NPS ARCHIVE
1997.06
MIRANDA, G.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INTERPOLATION WEIGHTS OF ALGEBRAIC MULTIGRID

by

Gerald N. Miranda, Jr.

June 1997

Thesis Advisor:	Van Emden Henson
Second Reader:	Christopher L. Frenzen

Thesis
M6326

Approved for public release; Distribution is unlimited.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, Va 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE June, 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE INTERPOLATION WEIGHTS OF ALGEBRAIC MULTIGRID	5. FUNDING NUMBERS
6. AUTHORS MIRANDA, GERALD N., JR.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT(*maximum 200 words*)

Algebraic multigrid (AMG) is a numerical method used to solve particular algebraic systems, and interest in it has risen because of its multigrid-like efficiency. Variations in methodology during the interpolation phase result in differing convergence rates. We have found that regular interpolation weight definitions are inadequate for solving certain discretized systems so an iterative approach to determine the weights will prove useful. This iterative weight definition must balance the requirement of keeping the interpolatory set of points "small" in order to reduce solver complexity while maintaining accurate interpolation to correctly represent the coarse-grid function on the fine grid. Furthermore, the weight definition process must be efficient enough to reduce setup phase costs.

We present systems involving matrices where this iterative method significantly outperforms regular AMG weight definitions. Experimental results show that the iterative weight definition does not improve the convergence rate over standard AMG when applied to M-matrices; however, the improvement becomes significant when solving certain types of complicated, non-standard algebraic equations generated by irregular operators.

14. SUBJECT TERMS Algebraic Multigrid, Matrix Equations, Interpolation Weights	15. NUMBER OF PAGES 88
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

Approved for public release; distribution is unlimited

INTERPOLATION WEIGHTS OF ALGEBRAIC MULTIGRID

Gerald N. Miranda, Jr.
Lieutenant, United States Navy
B.A., University of California, San Diego, 1990

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1997

NPS ARCHIVE
1997.06
MIRANDA, G.

~~Thesis~~
~~M6[illegible]~~
~~c.2~~

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY [illegible] 5101

ABSTRACT

Algebraic multigrid (AMG) is a numerical method used to solve particular algebraic systems, and interest in it has risen because of its multigrid-like efficiency. Variations in methodology during the interpolation phase result in differing convergence rates. We have found that regular interpolation weight definitions are inadequate for solving certain discretized systems so an iterative approach to determine the weights will prove useful. This iterative weight definition must balance the requirement of keeping the interpolatory set of points "small" in order to reduce solver complexity while maintaining accurate interpolation to correctly represent the coarse-grid function on the fine grid. Furthermore, the weight definition process must be efficient enough to reduce setup phase costs.

We present systems involving matrices where this iterative method significantly outperforms regular AMG weight definitions. Experimental results show that the iterative weight definition does not improve the convergence rate over standard AMG when applied to M-matrices; however, the improvement becomes significant when solving certain types of complicated, non-standard algebraic equations generated by irregular operators.

TABLE OF CONTENTS

I. INTRODUCTION **1**

II. THE FUNDAMENTALS OF MULTIGRID **5**

- A. THE PROBLEM STATEMENT 5
- B. NOTATION 6
- C. ITERATIVE METHODS 7
  - 1. The Weighted Jacobi Iteration on the "Model" Problem . 8
  - 2. Frequency Modes of the Error 8
- D. THE METHOD OF COARSE-GRID CORRECTION 12
  - 1. The Restriction Operator 12
  - 2. The Interpolation Operator 14
  - 3. The Algorithm of CG 17
- E. THE MULTIGRID V-CYCLE SCHEME 18
- F. THE STRATEGY OF NESTED ITERATION 21
- G. THE FULL MULTIGRID V-CYCLE 22
- H. IS MULTIGRID ENOUGH? 24

III. ALGEBRAIC MULTIGRID **27**

- A. WHY ALGEBRAIC MULTIGRID? 28
- B. APPLICATIONS OF AMG 29
- C. THE STEPS OF AMG 30
  - 1. The Setup Phase 30
  - 2. The Solution Phase 31
- D. DEFINING THE "GRID" 31
- E. CONNECTIONS AND CONVERGENCE 31

IV. CONSTRUCTING THE INTERPOLATION OPERATOR . . **35**

- A. ALGEBRAIC SMOOTHNESS 35
- B. INTERPOLATION ALONG DIRECT CONNECTIONS 36

C. THE ITERATIVE WEIGHT DEFINITION 37

D. INTERPOLATION CONSTRUCTION USING THE ITERATIVE WEIGHT DEFINITION 39

1. Initialization 41

2. Calculation 41

E. INTERPRETATION OF THE ALGORITHM 43

V. NUMERICAL RESULTS 45

A. ISOTROPIC PROBLEMS 46

B. THE ANISOTROPIC PROBLEM USING THE FINITE DIFFERENCE METHOD 48

C. THE ANISOTROPIC PROBLEM USING THE FINITE ELEMENT METHOD 50

D. PROBLEMS WITH COMPLEX DOMAINS 51

1. The Meshes 51

2. Results of Complex Domain Problems 54

E. COMPLEX DOMAINS AND OPERATORS 55

F. CONCLUDING REMARKS 58

VI. FUTURE RESEARCH 61

A. INTERPOLATION USING EIGENVECTORS 61

B. THE LEAST-SQUARES IDEA 62

C. THE COMPOSITE GRID FORMULATION 64

GLOSSARY 67

LIST OF REFERENCES 69

INITIAL DISTRIBUTION LIST 71

LIST OF FIGURES

1. Rectangular domain for discretization of a 2-D problem. 2
2. Discretized domain of the one-dimensional problem. 5
3. Fourier modes with wave-numbers $k = 1,\ 6,\ 12$. 9
4. Application of the weighted Jacobi iteration with $\omega = 2/3$ on Fourier modes with wave-numbers $k = 1,\ 3,\ 6,\ 12$. 10
5. Application of the weighted Jacobi iteration with $\omega = 2/3$ on the 1-D "model" problem. 11
6. Restriction by injection on a discretized sine wave. 14
7. Restriction by full weighting on an arbitrary vector. 15
8. Linear interpolation of a discretized cosine wave. 16
9. The multigrid V-cycle. 18
10. Size of the problem verses number of V-cycles performed. 20
11. The full multigrid V-cycle. 23
12. Relationship of fine grid to coarse grid points in a matrix. 38
13. Illustration of the neighborhood (N_i) of a fine grid point i. 40
14. Mesh 1. 51
15. Mesh 2. 52
16. Mesh 3. 52
17. Mesh 4. 53
18. Mesh 5. 53
19. Mesh 6. 54
20. The sparsity pattern of mesh 3. 56
21. Location of positive and negative entries of mesh 3. 57

LIST OF TABLES

I. Convergence rates of isotropic problems discretized using the finite difference method. 48

II. The anisotropic problem of the Laplacian operator discretized using the finite difference method. 49

III. The anisotropic problem with zero RHS discretized using the finite element method. 50

IV. Convergence rates of problems high in complexity using the standard Laplacian operator discretized using the finite element method. 54

V. Convergence rates of complex problems using an irregular operator discretized using the finite element method with $\gamma = 0.25$. 55

VI. Convergence rates of complex problems using an irregular operator discretized using the finite element method with $\gamma = 0.5$. 56

VII. Convergence rates over local refinements on mesh 3 discretized using the finite element method. 57

VIII. Convergence rates over local refinements on mesh 5 discretized using the finite element method. 58

LIST OF ACRONYMS AND SYMBOLS

AMG	Algebraic multigrid
CG	Coarse-grid correction scheme
FMG	Full multigrid (nested iteration)
FMV	Full multigrid with V-cycle
IWD	Iterative Weight Definition
MG	Multigrid
NI	Nested iteration
RHS	Right-hand side

∇^2	Laplacian operator; $\nabla^2 = (\frac{\partial}{\partial x})^2 + (\frac{\partial}{\partial y})^2 + (\frac{\partial}{\partial z})^2$ in Cartesian coordinates
Δ	same as ∇^2
Ω	the domain (grid) on which the problem is defined
Ω^h	the fine grid or the set of fine level variables $u^h_{i(k)}$ (in AMG), $\Omega^h = C^h \cup F^h$
Ω^H	the coarse grid or the set of coarse level variables u^H_k (in AMG)
$\partial\Omega$	the boundary of the domain
a_{ij}	the diagonal element of the matrix A in the i^{th} row
a_{ij}	the component of the matrix A in the i^{th} row and the j^{th} column
C	the set of points on the coarse grid
C_i	the set of interpolatory coarse-grid points for point i
δ_{ik}	the Kronecker-delta symbol, $\delta_{ik} = 0$ for $i \neq k$ and $\delta_{ik} = 1$ for $i = k$
e	the error vector
F	the set of points on the fine grid
h	the finer level (of two consecutive levels)
H	the coarser level (of two consecutive levels)
N^h_i	the set of points $\{j \in \Omega^h \mid j \neq i,\ a^h_{ij} \neq 0\}$ in the neighborhood of a point $i \in \Omega^h$
ρ	the asymptotic V-cycle convergence factor
r	the residual vector, $r = Ae$
σ^Ω	grid complexity; the ratio of the total number of points on all grids to that on the fine grid
σ^A	operator complexity; the ratio of the total number of nonzero entries in all the matrices to that in the fine-grid matrix
u	or u^h, the discretized solution vector
v	or v^h, the approximation vector to u
w_{ik}	the interpolation weights $= \eta_i \left(\lvert a^h_{ij} \rvert + \sum_{j \in D_i} \eta_i^{(j)} \lvert a^h_{ij} \rvert \right) \quad (i \in F, k \in C_i)$

ACKNOWLEDGMENTS

I would like to thank John Ruge for his insight into this problem of interpolation weights. His helpful suggestions while we were at the Copper Mountain conference were instrumental for me not only in understanding the problem, but moreso, in widening my avenues of research. His co-authored paper, *Algebraic Multigrid*, was certainly a catylst for my work.

Special thanks goes to Bill Briggs, to whom I owe a great deal even though we've never met. His book, *A Multigrid Tutorial*, is, in a sense, "the bible of multigrid" for the spirited student and layman alike. It played an integral role in my learning of the subject area as well as provided me with the necessary detail for my background work in multigrid methods. His presentation of the material could not have been better.

Finally, I extend a very personal thanks and a debt of gratitude to Van Emden Henson, my advisor, for stimulation in the subject, for informative discussions and mostly for a never ending supply of encouragement, especially when my work seemed to lead to nowhere in particular. Without his direction and availability for discussion, I certainly would have run about in circles. I cannot say enough about his countless hours of help in refreshening my memory when it seemed to go blank. Without him, this thesis would not have been possible.

I. INTRODUCTION

Precursors to multigrid (MG) methods were developed in the mid 1960's by Fedorenko (1964), and Bachvalov (1966), and remained fairly unknown until Achi Brandt [Ref. 1] fully developed multigrid in his 1973 article. In that paper, he demonstrated for the first time the utility of true multigrid methods as a fast numerical solvers for boundary value problems. An explosion of papers ensued and the 1980's produced a plethora of articles as well as efficient and reliable computer algorithms to further demonstrate the practicality and usefulness of MG. Confidence in multigrid grew when the true merits of this solver came to bear fruit in faster convergence rates over conventional methods. Proofs of these faster convergence rates coupled with a sufficient number of satisfactory numerical results gave way to a general acceptance of the method even to the most sceptic [Ref. 2].

Originally developed to solve simple boundary value problems, multigrid methods gained wide recognition for their speed and efficiency in solving general linear partial differential equations (PDEs) of the elliptic form, e.g., (in two dimensions)

$$-\nabla^2 u(x,y) = f(x,y),$$

or explicitly as

$$-u_{xx} - u_{yy} = f(x,y).$$

If this equation is now discretized, for example by finite differences, on some rectangular domain, $\Omega \subset \mathcal{R}^2$ as in Figure 1, and if we denote the boundary of the domain as $\partial\Omega$, then multigrid will solve problems involving Laplace's equation

$$\left.\begin{aligned} -\nabla^2 u = 0 \quad &\text{in} \quad \Omega \\ u = 0 \quad &\text{on} \quad \partial\Omega \end{aligned}\right\}, \tag{I.1}$$

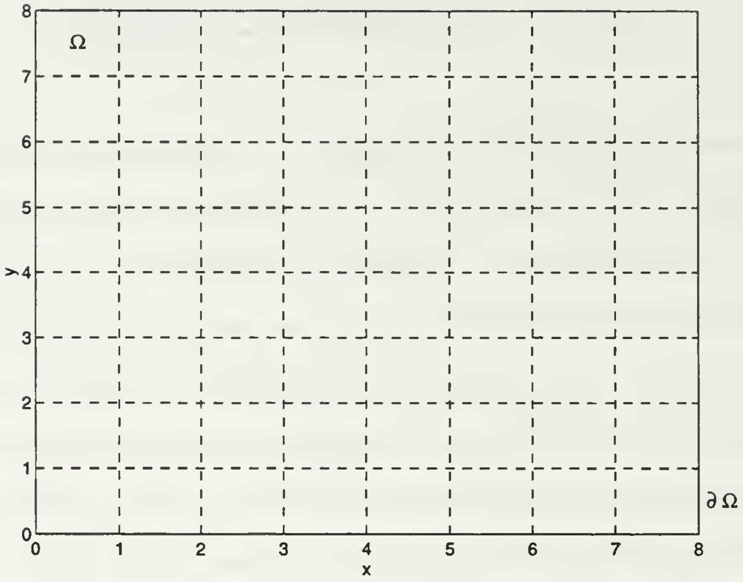

Figure 1. *Rectangular domain for discretization of a 2-D problem.*

Poisson's equation

$$\left.\begin{array}{l} -\nabla^2 u = S \quad \text{in} \quad \Omega \\ u = 0 \quad \text{on} \quad \partial\Omega \end{array}\right\}, \tag{I.2}$$

Helmholtz's equation

$$\left.\begin{array}{l} -\nabla^2 u + \kappa^2 u = S \quad \text{in} \quad \Omega \\ u = 0 \quad \text{on} \quad \partial\Omega \end{array}\right\}, \tag{I.3}$$

or even the anisotropic Poisson equation

$$\left.\begin{array}{l} -\varepsilon\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} = S \quad \text{in} \quad \Omega \\ u = 0 \quad \text{on} \quad \partial\Omega \end{array}\right\}, \tag{I.4}$$

in any number of dimensions almost effortlessly (comparatively speaking, of course).

These types of equations frequently arise in physical applications such as steady state temperature problems, fluid flow, orbital mechanics (gravitational fields), steady state electrical field problems, and many others. When a matrix equation arises from the discretization of (I.1) through (I.4), multigrid methods are quite successful.

MG methods are fast iterative solvers using a hierarchy of levels, or grids, and may be used with many types of discretization techniques such as finite difference or finite element methods. In solving problems of the elliptic nature using these discretization techniques, multigrid has proven to be the fastest numerical solution technique in the field. Unlike other numerical solvers, multigrid is general enough so that it can effectively use fairly arbitrary regions and boundary conditions and more importantly, does not depend on the separability of the differential equations.

In fact, from presentations at the Copper Mountain Conference on Multigrid Methods and from various papers which can be found on MG Net ("http://casper.cs.yale.edu/mgnet/www/mgnet-papers.html# Y"), we know that multigrid can also be directly applied to more complicated, non-symmetric and nonlinear systems of equations, like the Lame-System of elasticity or the Stokes (or Navier-Stokes) equations. Multigrid has been applied successfully to electrostatic and magnetostatic problems [Ref. 3], to statistical physics problems, to integral equations and to image reconstruction algorithms [Ref. 4, 5]. It can also be applied to problems in control theory, partical physics and permeable magnetic materials [Ref. 2].

The main concept of multigrid methods is "to complement the local exchange of information in point-wise iterative methods (on each level) by a global one utilizing several related systems, called *course levels*, with a smaller number of variables [Ref. 6]." The key to multigrid's performance then is to apply a relaxation technique as many times as needed to dampen the oscillatory modes of the error (since we know relaxation works best on highly oscillatory modes) and then apply a coarse-grid correction scheme to eliminate the smooth components of the error. This combination of relaxation and coarse-grid correction is the essence of why MG works so well.

Using multigrid methods, we can solve a large, difficult problem by reducing it iteratively into successive smaller, easier ones. At the coarsest level then the system can be solved directly using a known, efficient direct method (such as the LU-decomposition, Gaussian elimination, etc.). For systems such as (I.1) through (I.4), very efficient methods are available. From *A Multigrid Tutorial* [Ref. 7], we know that when applied to an $N \times N$ grid, multigrid methods are nearly optimal since they only require $O(N^2 \log N)$ arithmetic operations and hence approach the minimum operation count of $O(N^2)$ operations. In fact, it is shown in [Ref. 7] that the *full multigrid V-cycle* (FMV) method which we will discuss in the next chapter, requires only $O(N^d)$ to the level of discretization on the "model" problem (Laplace's equation (I.1)) of dimension d. This operation count is optimal.

II. THE FUNDAMENTALS OF MULTIGRID

There are two fundamental components of multigrid; one is the idea of *coarse-grid correction* (CG) and the other is that of *nested iteration* (NI). Presented here is a basic outline of both methods. To begin, we note that the principle concept of CG is that it is a correction strategy that transfers the components of the problem between the fine and coarse grids. Complementing coarse-grid correction is nested iteration which is based on the following concept: use the information on the coarse grids to provide an informed guess for the initial guess on the finer level.

A. THE PROBLEM STATEMENT

In order to understand the basics of multigrid and to motivate its usage, let us apply the method to a simple one-dimensional problem. We begin by discretizing a problem of the form

$$-\frac{d^2u(x)}{dx^2} + \sigma u(x) = f(x), \qquad 0 < x < 1, \qquad \sigma \geq 0 \tag{II.1}$$

with Dirichlet boundary conditions ($u(x) = 0$ on $\partial\Omega$) by partitioning the domain, as in Figure 2, into $N + 1$ points:

$$x_j = jh, \quad \text{where} \quad j = 0, \ldots, N.$$

We make $h = 1/N$ of constant width throughout the interval. This creates a grid

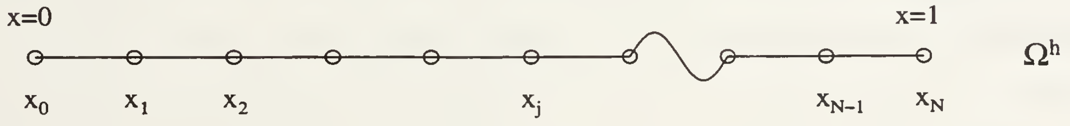

Figure 2. *Discretized domain of problem II.1.*

of $N - 1$ interior points. At each of these interior points, the equation in (II.1) is replaced by a second order discretization, such as a finite difference approximation

$$\begin{aligned} \frac{-v_{j-1}+2v_j-v_{j+1}}{h^2}+\sigma v_j = f_j, \qquad 1 \le j \le N-1 \\ v_0 = v_N = 0 \end{aligned} \tag{II.2}$$

where we define $f_j = f(x_j)$. The approximation (II.2) of (II.1) leads to a system of linear equations. In compact form, it is written as $Av = f$, or in explicit matrix notation, it is written

$$\frac{1}{h^2}\begin{bmatrix} 2+\sigma h^2 & -1 & & & \\ -1 & 2+\sigma h^2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+\sigma h^2 & -1 \\ & & & -1 & 2+\sigma h^2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ \\ v_{N-1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \\ f_{N-1} \end{bmatrix}$$

where the matrix A is tridiagonal, symmetric, positive definite and has dimension $N-1 \times N-1$. v is a column vector containing the approximate discretized solution of $u(x)$ and $f_i = f(x_i)$ is the discretized right-hand side (RHS).

B. NOTATION

Before we continue with the outline it is instructive to define a few terms. Let Ω^h denote the original grid or the domain on which the original problem is defined. We call Ω^h the *finest grid.* The symbol, Ω^{2h}, is defined to mean the *next coarser grid* (usually of size $N/2-1$ but certainly not restricted to that). Similarly, v^h will be an approximation to the exact solution, u^h, on Ω^h. This means that v^{2h} represents the approximate solution to u^{2h} on the next coarser level Ω^{2h}. This notation will be consistent for all such grids, Ω^{kh}. The algebraic error, e^{kh}, on Ω^{kh} ($k = 2^n$, $n = 0, 1, 2, 3, \ldots$) is given then by $e^{kh} = u^{kh} - v^{kh}$ where $k = 1$ represents the finest level.

When we translate the vectors only between two consecutive grids, we will use the notation Ω^h to represent the finer of the two grids and Ω^H to denote the coarser one. This makes it easier to follow the grid transfer sequence vice getting caught up in exponential notation of the varying grids. Now that we have dispensed with some of the notational formalities, let us move on to more of the important issues.

C. ITERATIVE METHODS

We introduce *iterative methods* here to illustrate the need for *coarse-grid correction.* First of all, iterative methods *are* needed and in fact, are required since the use of direct methods can be impractical if the matrix A is large and sparse. The sought-after factors of A using these direct methods can, and tend to, be very dense. Even when the matrix is banded (but still sparse), algorithms that are ideal for factoring such problems may be difficult to implement [Ref. 8]. Iterative methods by contrast "generate a sequence of approximate solutions and essentially involve the matrix A only in the context of matrix-vector multiplication [Ref. 8]." Iterative methods are attractive and easy to use because of their simplicity in implementation but may be prohibitively slow when the error is smooth.

What does it mean when we say, "when the error is smooth?" To answer this and to understand the need for coarse-grid correction, we must first show the power of iterative methods and then focus on their limitations. To avoid any confusion, we note here that the phrase "iterative method" is isomorphic to "relaxation scheme" or just simply, "relaxation." We demonstrate relaxation with the *weighted Jacobi* iteration since it is simple in nature and offers the same benefits as other iterative methods; in addition, it shows the common defects of relaxation in general. Other iterative methods include (regular) Jacobi, Gauss-Seidel, successive over-relaxation (SOR) and Chebyshev semi-iterative; of course, there are many others.

1. The Weighted Jacobi Iteration on the "Model" Problem

To begin, let us solve the following one-dimensional "model" problem which is problem (II.2) with σ and f set to zero. The problem then becomes

$$\begin{gathered} -u_{j-1} + 2u_j - u_{j+1} = 0, \qquad 1 \le j \le N-1 \\ u_0 = u_N = 0. \end{gathered} \tag{II.3}$$

We solve (II.3) by using the weighted Jacobi iteration. This iteration is computed using

$$v_j^{(\text{new})} = (1-\omega)v_j^{(\text{old})} + \omega v_j^*, \qquad 1 \le j \le N-1$$

where

$$v_j^* = \frac{1}{2}\left(v_{j-1}^{(\text{old})} + v_{j+1}^{(\text{old})} + h^2 f_j\right), \qquad 1 \le j \le N-1$$

and $\omega \in \mathcal{R}$ is a weighting factor chosen such that $0 \le \omega \le 1$. In our use of the weighted Jacobi iteration, we set $\omega = 2/3$. It turns out to be the optimal weighting factor.

For the moment, let us assume that we have found an approximation, $v \neq 0$, to the discretized solution, $u = 0$, on our model problem. Since the system is homogeneous and linear, we know the error, e, exactly, namely $-v$ since $e = u - v = -v$. In reality, the error will consist of many different frequencies but for simplicity, let us assume that e consists of only three frequency modes: one high, one low and a third mode in the middle. Our assumption using the three modes here is merely to amplify the benefits of relaxation while simultaneously exploiting the inherent defects.

2. Frequency Modes of the Error

Consider an initial approximation to the solution consisting of the Fourier modes

$$v_j = \sin\left(\frac{jk\pi}{N}\right)$$

where $0 \le j \le N$ and $1 \le k \le N-1$. The number k represents the wave number of the Fourier mode. The kth mode consists of $k/2$ full sine waves on the interval.

As we will see shortly, the wave number k (frequency mode) will play a major role in the overall effect of relaxation. Figure 3 illustrates the modes $v_j = \sin\left(\frac{jk\pi}{N}\right)$ with wave-numbers $k = 1,\ 6,\ 12$ on a grid with $N = 64$. Wave number $k = 1$ represents the lowest frequency while $k = 63$ is the highest one. We now iterate (or "relax") on problem (II.3) by applying the weighted Jacobi iteration with $\omega = 2/3$ on a grid of size $N = 64$.

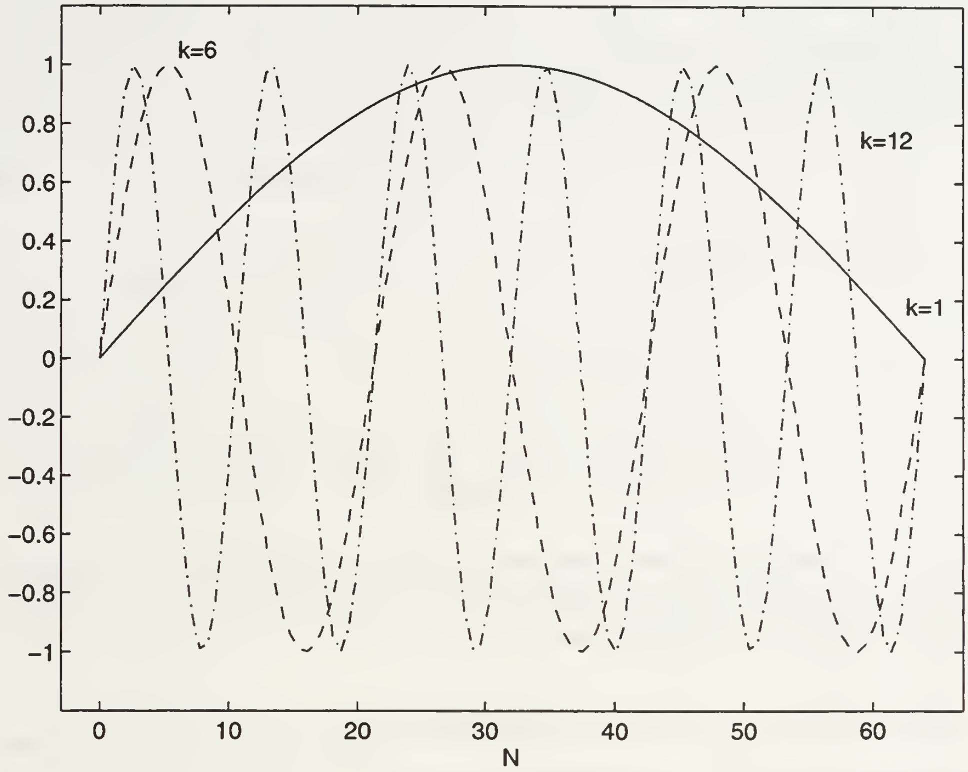

Figure 3. *Fourier modes* $v_j = \sin\left(\frac{jk\pi}{N}\right)$ *with wave-numbers* $k = 1,\ 6,\ 12$ *on a grid with* $N = 64$. *The kth mode consists of* $k/2$ *full sine waves on the interval.* v_1 *is represented by the solid line,* v_6*, the dashed and* v_{12}*, the dot-dashed.*

But wait! Maybe we are being a bit too hasty. Let's step back from solving the problem for a moment. Before we see what happens with our mixed-frequency initial guess, it is instructive to first watch what happens to the individual components in our initial guess. Figure 4 shows the application of the weighted Jacobi iteration metho on problem (II.3) with v_1, v_3, v_6, and v_{12} as separate initial guesses.

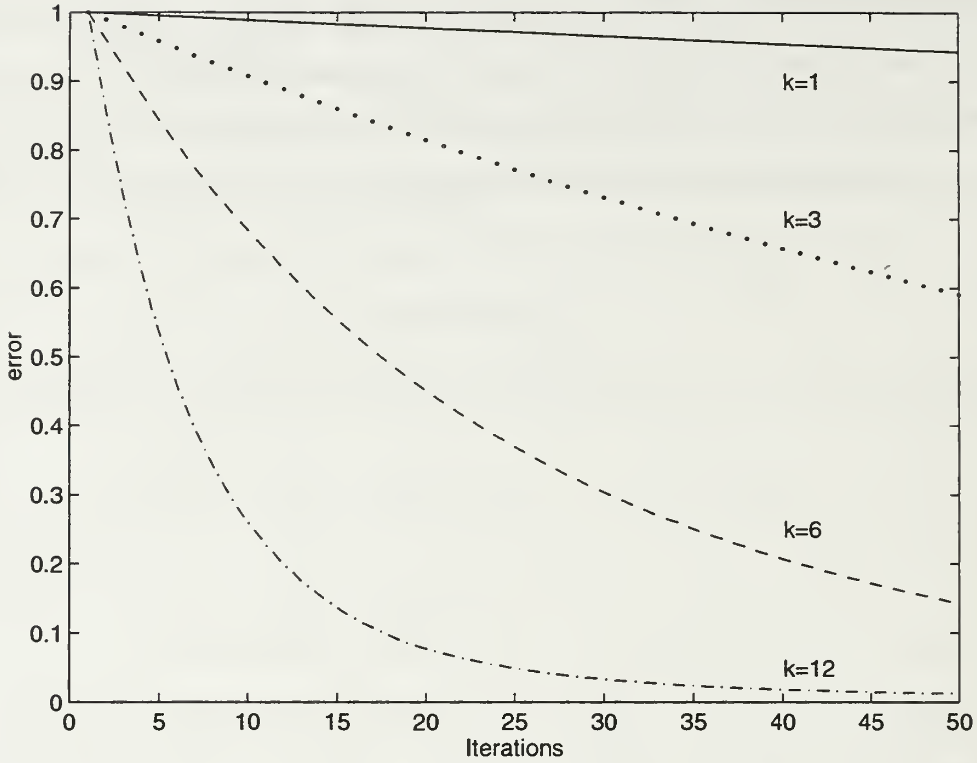

Figure 4. *The weighted Jacobi iteration with* $\omega = 2/3$ *applied to the 1-D "model" problem with* $N = 64$ *and with initial guesses* v_1, v_3, v_6, v_{12}. $||e||_\infty$ *is plotted against the iteration number for 50 iterations.*

Notice that the error quickly decreases to zero within a few iterations for the high- frequency mode (v_{12}) and although not quite as rapidly, the error also decreases for the middle-frequency mode (v_6) but it doesn't quite go to zero. In extreme contrast, the low-frequency modes (v_1 and v_3) appear to be relatively untouched by the relaxation scheme. In fact, a prohibitively large number of iterations will still not effectively reduce the error. It is the wonderful property of iterative methods which allow for the quick elimination of high-frequency modes. On the other hand, they are quite defective in decreasing the error when it consists solely of low-frequency modes.

We now return to problem (II.3) with our mixed-frequency initial guess. It is of the form

$$v_j = \frac{1}{3}\left[\sin\left(\frac{j\pi}{N}\right) + \sin\left(\frac{6j\pi}{N}\right) + \sin\left(\frac{12j\pi}{N}\right)\right].$$

Figure 5 shows how the error of a typical approximation vector decreases using relaxation alone. It is not uncommon that the error decreases rapidly, usually within a few iterations. This is due to the quick elimination of the high-frequency modes. But after this initial rapid decrease, the error is reduced much more slowly. The main culpritsare the low-frequency modes. "The important observation is that the standard iterations converge very quickly as long as the error has high-frequency components. However, the slower elimination of the low-frequency components degrade the performance of the methods [Ref. 7]."

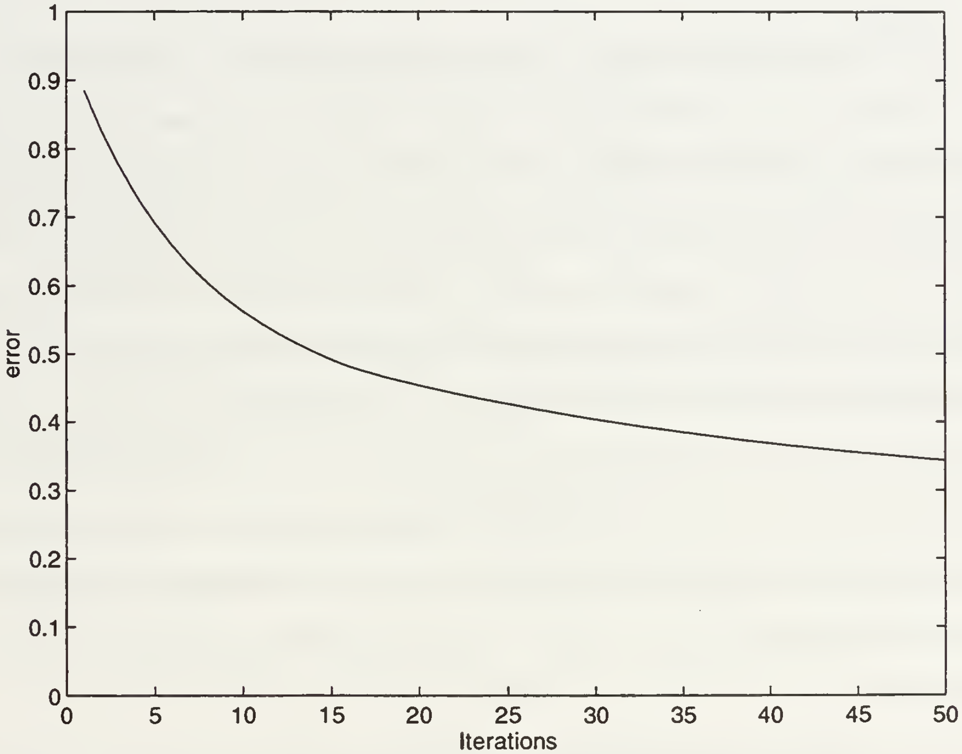

Figure 5. *The weighted Jacobi iteration with* $\omega = 2/3$ *applied to problem (II.3) with* $N = 64$ *and with the initial guess* $v_j = \frac{1}{3}\left[\sin\left(\frac{j\pi}{N}\right) + \sin\left(\frac{6j\pi}{N}\right) + \sin\left(\frac{12j\pi}{N}\right)\right]$. $||e||_\infty$ *is plotted against the iteration number for 50 iterations.*

What we have seen is that iterative methods work very well for the first several iterations but inevitably the convergence rate slows and the methods seem to stall. This phenomenon is due to the fact that the rapid decrease in error in the early stages of the method is from the swift, efficient elimination of the oscillatory modes (high-frequency components). However, once the high-frequency components have been removed from the error, the iteration is less effective in reducing the remaining smooth modes (low-frequency components).

This property of quickly removing the oscillatory modes but leaving the smooth modes of the error is called the *smoothing property.* The smoothing property of iterative methods (relaxation) is a serious limitation. How do we overcome this grave obstacle? It is in the application of coarse-grid correction to the remaining "smooth error" that we remedy this problem.

D. THE METHOD OF COARSE-GRID CORRECTION

In light of the limitations of iterative methods, we seek a post-relaxation scheme. The purpose of the coarse-grid correction scheme (CG) is to eliminate the low-frequency modes of the error once relaxation has become ineffective. The means by which we do this is intergrid transfers. Intergrid transfers move vectors (and the matrix) from the fine grid to the coarse grid and vice-versa by means of restriction and interpolation operators. In "moving" to a coarser grid, we observe that the smooth modes of the error become higher-frequency modes on this new grid. On the coarse-grid now, the error has oscillatory components again; but we can effectively remove those modes using relaxation. This two step methodology is the basis for CG. To understand what it means to "move" a vector between grids, we introduce a few required operators.

1. The Restriction Operator

Starting on the fine grid, Ω^h, we move the problem $A^h v^h = f^h$ to the next coarser grid, Ω^H, by applying an intergrid transfer to A^h, v^h and f^h. This operation gives us the coarse-grid equivalent to that problem but on Ω^H: $A^H v^H = f^H$. In doing this though, we must be careful to ensure that the coarse-grid problem "be consistent with the differential problem in the same way as the fine-grid problem [Ref. 2]." The action of moving to the coarse-grid problem is performed by the *restriction operator*

$$I_h^H : \mathcal{R}^{N-1} \longrightarrow \mathcal{R}^{\frac{N}{2}-1}.$$

I_h^H is a linear operator from $\mathcal{R}^{N-1}$ to $\mathcal{R}^{\frac{N}{2}-1}$ and has rank $N/2-1$ (see [Ref. 2, 7, 9] for more on intergrid transfers). The most obvious of these types of restriction operators is *injection.* Injection is a linear operator which produces the coarse-grid solution approximation

$$v^H = I_h^H v^h$$

where

$$v_j^H = v_{2j}^h \quad \text{for} \quad 1 \le j \le \frac{N}{2} - 1.$$

In words, injection gets the value of the coarse-grid vector directly from the associated fine grid point. Figure 6 shows how the injection operator acts on a discretized sine wave when moving the vector from Ω^h to Ω^H.

Another type of restriction operator is *full weighting.* It is also a linear operator from $\mathcal{R}^{N-1}$ to $\mathcal{R}^{\frac{N}{2}-1}$ with rank $N/2-1$. Like injection, it produces the coarse-grid solution approximation

$$v^H = I_h^H v^h$$

but in this case the transfer to the coarse grid is a bit different:

$$v_j^H = \frac{1}{4}\left(v_{2j-1}^h + 2v_{2j}^h + v_{2j+1}^h\right) \quad \text{for} \quad 1 \le j \le \frac{N}{2} - 1.$$

In the case of the full weighting restriction operator, "the values of the coarse grid vectors are a weighted average of values at neighboring fine grid points [Ref. 7]." Figure 7 shows how the full weighting restriction operator acts upon a vector when moving the vector from the fine grid to the coarse grid.

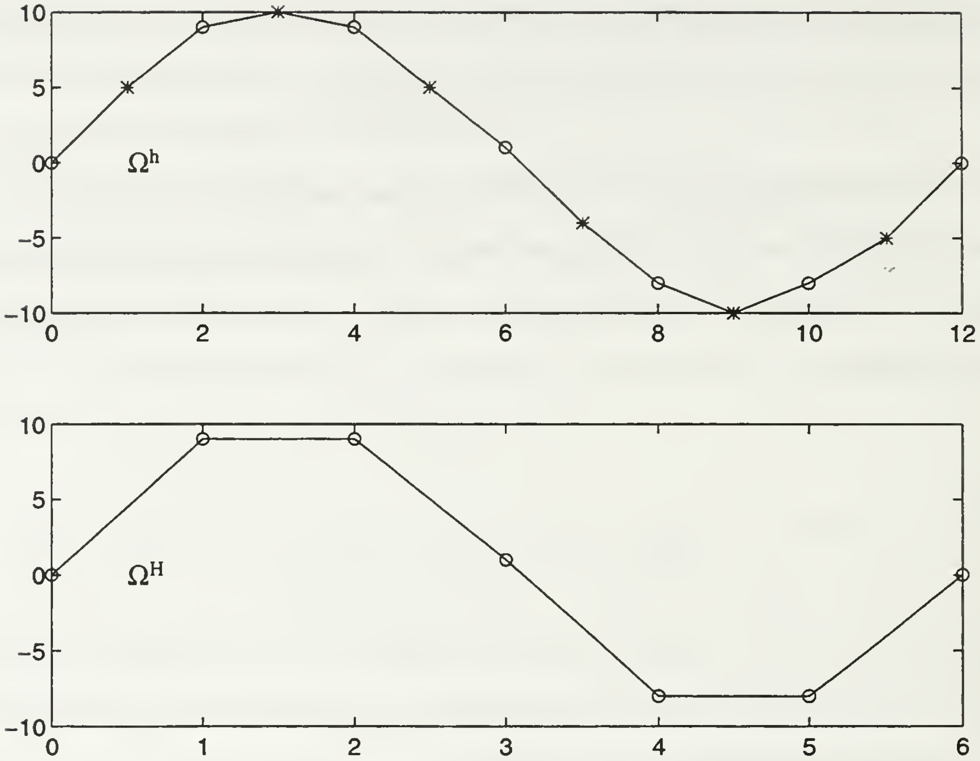

Figure 6. *Restriction by injection on a discretized sine wave from Ω^h to Ω^H. The circles (o) on Ω^h directly become the coarse-grid points on Ω^H.*

2. The Interpolation Operator

Once on the coarse grid, Ω^H, and after we complete our calculations there, we need to be able to return to Ω^h. We introduce the tool that will assist us in this task: the interpolation operator

$$I_H^h : \mathcal{R}^{\frac{N}{2}-1} \longrightarrow \mathcal{R}^{N-1}.$$

The interpolation operator is a linear operator from $\mathcal{R}^{\frac{N}{2}-1}$ to $\mathcal{R}^{N-1}$ and has full rank. I_H^h moves the coarse-grid vectors up one level by the calculation

$$v^h = I_H^h v^H$$

where

$$v_{2j}^h = v_j^H \quad \text{for} \quad 0 \le j \le \frac{N}{2} - 1.$$

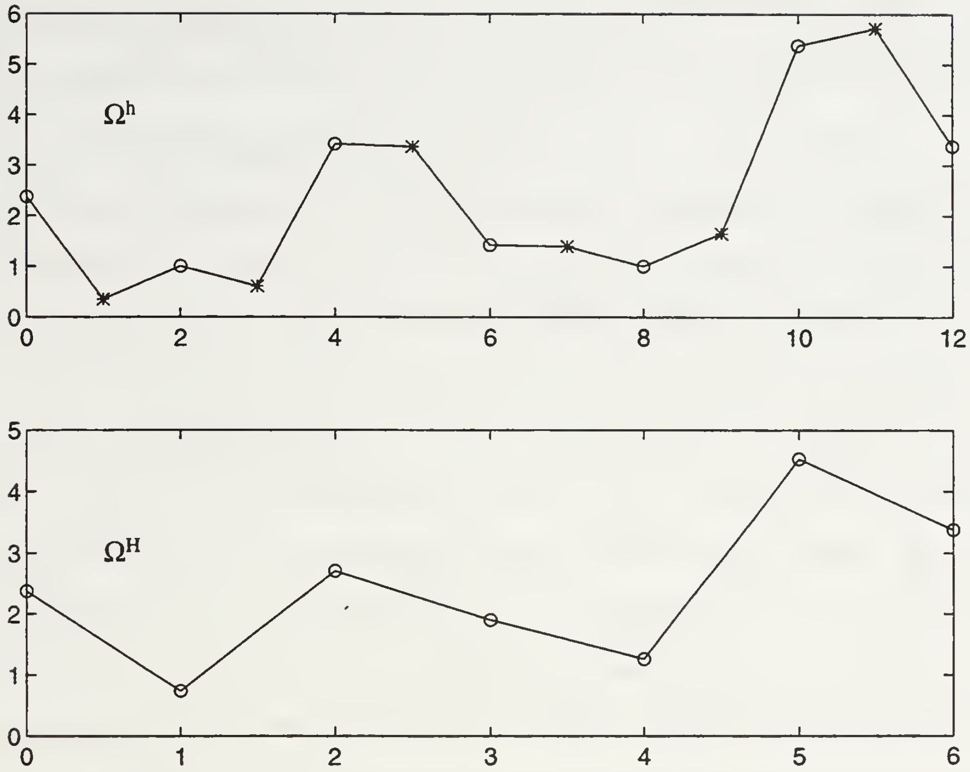

Figure 7. *Restriction by full weighting on a arbitrary vector from* Ω^h *to* Ω^H. *The circles (o) on* Ω^H *are the coarse-grid points and are weighted averages of neighboring fine grid point values.*

The interpolation operator produces a vector on the fine grid that is a weighted average of its adjacent points on the coarse grid. Figure 8 graphically depicts the action of I_H^h on a discretized cosine wave.

Now that we have our intergrid transfer operators defined, it makes sense to talk about moving the matrix between levels. On Ω^h, $A^h = A$. But what is A^H and how do we get it? As a definition, we will call A^H the Ω^H version of A^h or, the *coarse-grid operator.* We get A^H from A^h by the calculation

$$A^H = I_h^H A^h I_H^h, \tag{II.4}$$

and we recover A^h from A^H by the calculation

$$A^h = I_H^h A^H I_h^H.$$

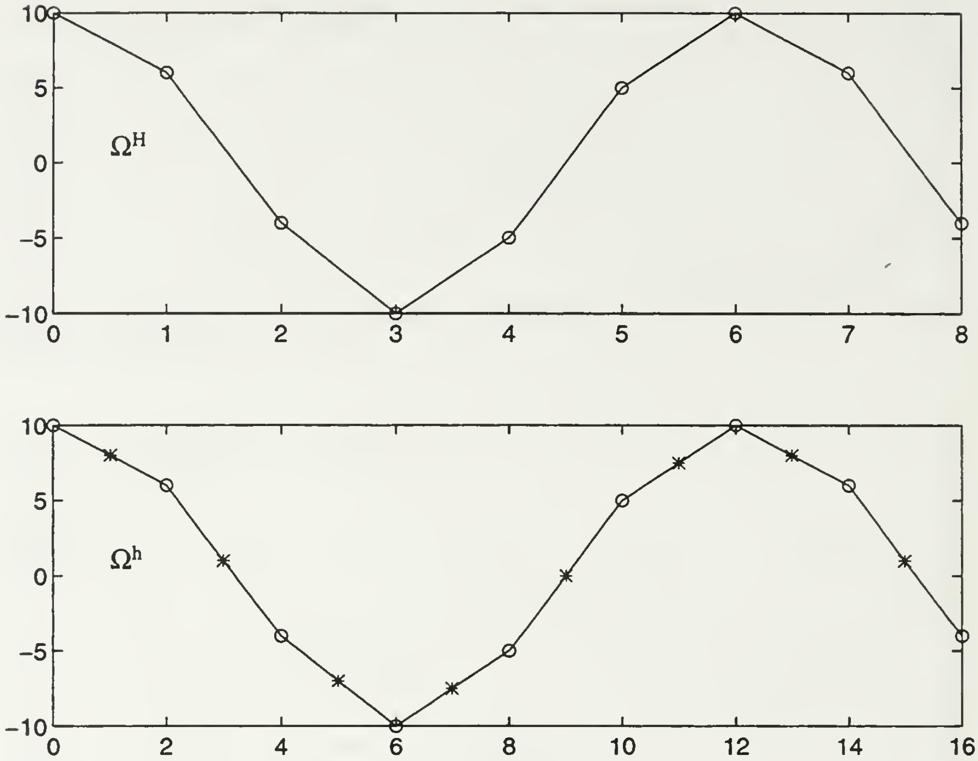

Figure 8. *Interpolation of a cosine wave from Ω^H to Ω^h. The stars $(*)$ on Ω^h are the interpolated values from their adjacent coarse-grid points on Ω^H.*

This is how we move the operator A on all levels. The coarse-grid operator (II.4) is called the *Galerkin condition.* A second special and important relationship between the full weighting restriction operator and the linear interpolation operator is

$$I_H^h = c\left(I_h^H\right)^T \quad \text{where} \quad c \in \mathcal{R}; \tag{II.5}$$

that is, they are transposes of each other up to a (real) constant which turns out to be essential. (II.4) along with (II.5) are called the *variational properties.*

We note here that there is a disadvantage in using a linear interpolation operator. In fact, Wesseling explains:

> Because of the arbitrariness in choosing the direction of the diagonals of the interpolation triangles, there may be a loss of symmetry, that is, if the exact solution of a problem has a certain symmetry, it may happen that the numerical solution does not reproduce this symmetry exactly, but only with truncation error accuracy. [Ref. 2]

However, linear interpolation is much cheaper (and easier) to use than, for example, bilinear or quadratic interpolation. So in our work, we only use linear interpolation.

3. The Algorithm of CG

To begin the *coarse-grid correction* algorithm, we perform a relaxation scheme (e.g., Gauss-Seidel or Jacobi) on Ω^h on the original equation $A^h u^h = f^h$ with an arbitrary initial guess v^h. This has the effect of producing a much better guess since it removes the oscillatory components of the error right from the start. We now compute the residual vector, r^H, on the coarser grid $r^H = I_h^H(f^h - A^h v^h)$ and then solve $A^H e^H = r^H$ for the algebraic error, e^H on Ω^H. Now that the error e^H is computed, we return to Ω^h by using the linear interpolation operator, I_H^h, defined above. Since the error was smooth on Ω^H, I_H^h transfers that error very accurately back to Ω^h. We now correct the fine grid approximation $v^h \longleftarrow v^h + I_H^h e^H$ by adding the interpolated error e^h $(= I_H^h e^H)$ back into the initial guess. Since $u^h = v^h + e^h$, we have improved our fine grid approximation to the solution and in theory, we have found the solution. To improve it further, we perform relaxation on the original equation $A^h u^h = f^h$ with the updated guess v^h to obtain a final approximation to the exact discretized solution u^h. The coarse-grid correction scheme (in compact form) is printed below and is reproduced from Briggs', *A Multigrid Tutorial* [Ref. 7].

```
$v^h \longleftarrow CG(v^h, f^h)$

Relax $\nu_1$ times on $A^h u^h = f^h$ on $\Omega^h$ with initial guess $v^h$.
Compute $r^H = I_h^H(f^h - A^h v^h)$.
    Solve $A^H e^H = r^H$ on $\Omega^H$.
Correct the fine grid approximation: $v^h \longleftarrow v^h + I_H^h e^H$.
Relax $\nu_2$ times on $A^h u^h = f^h$ on $\Omega^h$ with (updated) initial guess $v^h$.
```

Here, ν_1 and ν_2 are small positive integers, usually between one and three.

E. THE MULTIGRID V-CYCLE SCHEME

The multigrid V-cycle scheme is a recursive algorithm that iteratively imbeds the two level CG algorithm within itself. The physical property of this scheme (the pattern it creates) is how it gets its name. Figure 9 shows a graphic illustration of the V-cycle pattern.

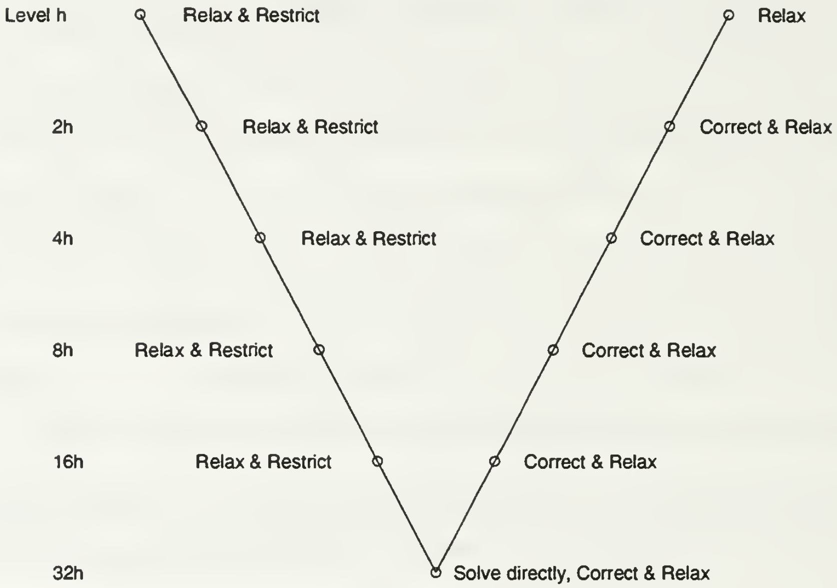
Level h
Relax & Restrict
Relax
2h
Relax & Restrict
Correct & Relax
4h
Relax & Restrict
Correct & Relax
8h
Relax & Restrict
Correct & Relax
16h
Relax & Restrict
Correct & Relax
32h
Solve directly, Correct & Relax

Figure 9. *The multigrid V-cycle.*

We start one CG algorithm on the finest level and a new one on each successive level all the way down to the coarsest level where we the problem is solved directly. After we solve the problem on the coarsest level, we now complete each CG algorithm at each successive level all the way back to the finest level. To clarify what the V-cycle is really doing, we present the algorithm in a more structured format. In the code, it is assumed again that there are $Q > 1$ grids with the coarsest grid spacing given by Lh, where $L = 2^{Q-1}$. This procedure is also a reproduction from Briggs' tutorial [Ref. 7].

```
$v^h \longleftarrow V(v^h, f^h)$

Relax on $A^h u^h = f^h$ $\nu_1$ times with initial guess $v^h$.
Compute $f^{2h} = I_h^{2h} r^h$.
    Relax on $A^{2h} u^{2h} = f^{2h}$ $\nu_1$ times with initial guess $v^{2h} = 0$.
    Compute $f^{4h} = I_{2h}^{4h} r^{2h}$.
        Relax on $A^{4h} u^{4h} = f^{4h}$ $\nu_1$ times with initial guess $v^{4h} = 0$.
        Compute $f^{8h} = I_{4h}^{8h} r^{4h}$.
                  ⋮
                    Solve $A^{Lh} u^{Lh} = f^{Lh}$.
                  ⋮
        Correct $v^{4h} \longleftarrow v^{4h} + I_{8h}^{4h} v^{8h}$.
        Relax on $A^{4h} u^{4h} = f^{4h}$ $\nu_2$ times with initial guess $v^{4h}$.
    Correct $v^{2h} \longleftarrow v^{2h} + I_{4h}^{2h} v^{4h}$.
    Relax on $A^{2h} u^{2h} = f^{2h}$ $\nu_2$ times with initial guess $v^{2h}$.
Correct $v^h \longleftarrow v^h + I_{2h}^h v^{2h}$.
Relax on $A^h u^h = f^h$ $\nu_2$ times with initial guess $v^h$.
```

For all its simplicity, this algorithm is the basic structure of the multigrid methodology. It takes individual techniques and well known concepts (relaxation and intergrid transfers) inherent with individual defects and integrates them in a cohesive way, capitalizing on the benefits of each to produce an algorithm that is extremely efficient and quite powerful.

Given the recursive nature of the V-cycle algorithm above, we now present it in a more compact form.

```
$v^h \longleftarrow V(v^h, f^h)$

1. Relax $\nu_1$ times on $A^h u^h = f^h$ with a given initial guess $v^h$.
2. If $\Omega^h$ = the coarsest grid, then goto 4.
   Else $f^{2h} \leftarrow I_h^{2h}(f^h - A^h v^h)$
        $v^{2h} \leftarrow 0$;
        $v^{2h} \leftarrow V^{2h}(v^{2h},\ f^{2h})$.
3. Correct $v^h \leftarrow v^h + I_{2h}^h v^{2h}$.
4. Relax $\nu_2$ times on $A^h u^h = f^h$ with initial guess $v^h$.
```

The V-cycle is just one of many multigrid cycling schemes. To see its effectiveness, we show in Figure 10 how the V-cycle algorithm tailors itself to the problem vice the size of the matrix involved. The number of V-cycles required to solve the problem to truncation error barely grows even though we considerably increase the matrix size.

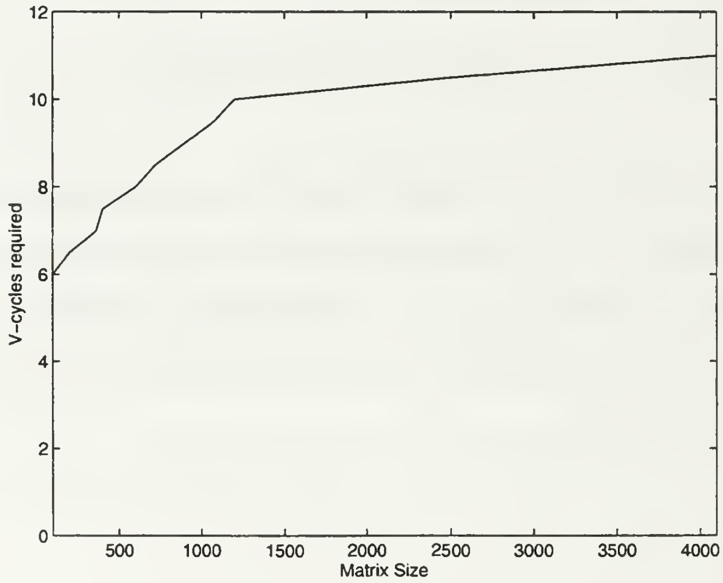

Figure 10. *Size of the problem verses number of V-cycles performed.*

F. THE STRATEGY OF NESTED ITERATION

When no information about the solution to the problem is known before-hand, it is computationally wasteful to begin solving the problem *on the finest grid* with just any initial random guess, v^h. If a poor choice is made, the speed of convergence may be strongly affected. The algorithm may altogether fail to converge if constrained by the number of iterations especially when the problem is nonlinear. Computing the residuals on the "smaller" coarser grids is so much cheaper and it makes more sense than to remain only on Ω^h. Therefore, it is better to use the information of the coarse grids to provide an informed guess v^h on the finest grid. In addition to providing a better initial guess on Ω^h, nested iteration gives us a better choice for v^{kh} at each level k [Ref. 2]. This idea of using information inherent in the problem from the coarser grids to develop better initial guesses is the basis of *nested iteration.* In compact form, the NI algorithm is

$$v^h \longleftarrow NI(v^h, f^h)$$

```
For $k = 2, 4, 8, 16, \ldots, L$
    Compute $f^{kh} = I^{kh}_{(k-1)h} f^{(k-1)h}$.
end
Relax $\nu_1$ times on $Av = f$ on the coarsest level $\Omega^{Lh}$ and solve
    $A^{Lh} v^{Lh} = f^{Lh}$ for $v^{Lh}$.
Interpolate $v^{Lh}$ to obtain an initial guess for the $\Omega^{(L-1)h}$ problem.
For $k = L-1, \ldots, 8, 4, 2, 1$
    Compute $v^{(k-1)h} = I^{(k-1)h}_{kh} v^{kh}$.
    Relax $\nu_2$ times on $Av = f$ on $\Omega^{kh}$ to obtain an initial guess for the $\Omega^{(k-1)h}$
        problem.
end.
```

In this algorithm, ν_1 and ν_2 are again small positive integers and there are $Q > 1$ grids with the coarsest grid spacing given by Lh, where $L = 2^{Q-1}$.

G. THE FULL MULTIGRID V-CYCLE

In this section, we present a powerful algorithm that combines nested iteration with the V-cycle. We also talk about how it works and why it is necessary. The algorithm to which we are referring is called the *full multigrid V-cycle* (FMV) and it is the method which is computationally of optimal order. We already saw the algorithms for NI and the V-cycle and reasoned for their need. Now we join them into a single, efficient method.

In the FMV algorithm, the first approximation to the solution is obtained by interpolation of the solution from the next coarser grid, which has already been computed by another FMV algorithm. This is where NI plays its role. Once we achieve a first approximation, we then apply the multigrid V-cycle. "This combination should suffice to give a fine-grid solution to the level of truncation error [Ref. 10]."

The FMV algorithm in compact form is (reproduced from [Ref. 7]):

$$v^h \longleftarrow FMV^h(v^h,\ f^h)$$

```
1. If $\Omega^h$ = the coarsest grid, then goto step 3.
   else
      $f^{2h} \leftarrow I_h^{2h}(f^h - A^h v^h)$
      $v^{2h} \leftarrow 0;$
      $v^{2h} \leftarrow FMV^{2h}(v^{2h},\ f^{2h}).$
2. Correct $v^h \leftarrow v^h + I_{2h}^h v^{2h}.$
3. $v^h \leftarrow V^h(v^h,\ f^h)$   $\nu_0$ times.
```

Figure 11 shows how the FMV scheme operates over five grids. Notice that each V-cycle is proceeded by a smaller V-cycle which is designed to provide the best initial guess for the following cycle. Shortly, we will see that the extra work required in the preliminary V-cycle is not only of minimal cost, but it will generally pay for itself over the course in solving the entire problem [Ref. 7]. We also note here that FMG cycles are less sensitive than the multigrid cycles but with one cycle per refinement, we can still guarantee a solution to the level of truncation error [Ref. 10].

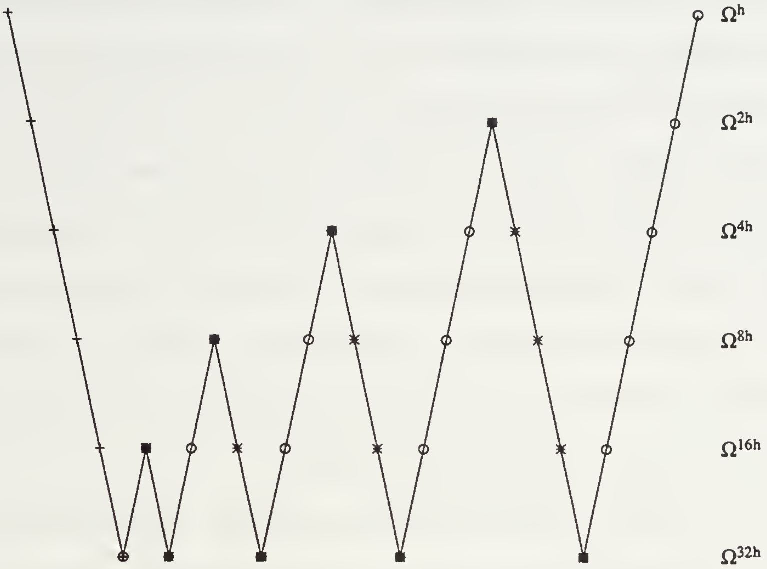

Figure 11. *The full multigrid V-cycle. The (∗) signify work performed before CG whereas (o) is the post coarse-grid correction work. The (+) on the first branch represent the restriction of f; no relaxation is performed here.*

Although the FMV algorithm costs more per cycle than the regular V-cycle, it is also more effective. “The key observation in the FMV argument is that before the Ω^h problem is even touched, the Ω^{2h} problem has already been solved to the level of truncation [Ref. 7].” The reason is that we now have a better initial guess for the *next* finer grid from our use of nested iteration. In fact, we know that because of this “extra” work during the preliminary cycling through the coarser grids, we only require $O(1)$ V-cycles by the time we finally reach the finest grid. So the computational cost of the FMV method applied to a d-dimensional problem, as mentioned before, is $O(N^d)$, which is optimal vice the cost of the V-cycle method alone which is of order $O(N^d \log N)$.

H. IS MULTIGRID ENOUGH?

The heart of multigrid intertwines relaxation methods such as Jacobi or Gauss-Seidel with intergrid operators like coarse-grid correction. In consideration of a general all-purpose elliptic equation solver, the two techniques enhance each other's effectiveness such that the overall algorithm – *multigrid* – is more robust and powerful than the sum of its parts. To be more specific, it is a known fact that relaxation effectively smoothes highly oscillatory modes of the error but it is ineffective on the smooth modes. This knowledge, coupled with the fact that any vector that lies in the range of the interpolation operator also lies in the null space of the coarse-grid correction operator, provides us a process that ensures that both the smooth and oscillatory modes of the error are suitably damped.

Multigrid methods have their limitations, though. Suppose we wish to solve a non-elliptically structured problem or even a regular elliptic problem but defined on an irregular grid. How, for example, can we apply multigrid methods to these types of systems. Will it even work at all? Can we tailor the irregularities in the meshes? How do we apply MG to problems where no geometric structure (or grid) exists? Some specialized multigrid codes have been written for specific, ill-structured problems to satisfy the irregular needs of the individual problem but that defeats the issue of providing a solver that is *more nearly* a "black-box" type of solver. Every specialization restricts the scope of the discretization method with which the multigrid code can be used [Ref. 11]. A significant effort may be necessary to prevent information loss in the discretization and still, it may be prone to errors. It is very difficult "to construct an efficient preconditioning method (which modifies the condition number of the matrix A) for arbitrary 'purely algebraic' problems [Ref. 9]."

What we really want to do is to solve purely algebraic problems with the same speed, efficiency and reliability that multigrid provides to geometrically structured problems. We require a method that "should use only information in the matrix of the system and as little extra information as possible [Ref. 11]." In other words, we want to extend all of the benefits of multigrid methods to a new type of solver, *algebraic multigrid.*

III. ALGEBRAIC MULTIGRID

Algebraic multigrid (AMG) methods are a required extension of the multigrid methodology. They were first introduced in the mid 1980's by A. Brandt [Ref. 12], S. McCormick and J. Ruge [Ref. 13] and further developed by J. Ruge and K. Stüben in their paper, *Algebraic Multigrid* [Ref. 14]. The methods arose from the need to solve purely algebraic systems (*matrix equations*) of the form

$$Au = f \quad \text{or} \quad \sum_{j=1}^{n} a_{ij}u_j = f_i \qquad (i = 1, \ldots, n) \tag{III.1}$$

with the "multigrid-style" cycling process without any underlying geometry. Although there is nothing special about the form in (III.1) as is, it is the form required for our work in AMG. In their paper, Ruge and Stüben show the usefulness of AMG when applied to a "model class" of matrices - *symmetric M-matrices.* (Recall that the matrix A is an M-matrix if $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$.) They also relaxed the assumption to weak diagonal dominance ($|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}|$, $i = 1, \ldots, n$) and showed uniform two-level convergence for that case as well. In addition, they imply that AMG may be attractive as a "black box" solver for algebraically posed elliptic problems as well as for certain other types of operators that generate matrices with similar characteristics.

Moreover, for particular types of matrix equations, algebraic multigrid is a robust and efficient matrix equation solver. It is designed to solve these types of matrix equations (III.1) using the same principles of standard multigrid methods, without requiring the need of an underlying geometry of the continuous problem. Application of AMG to many problems involving systems that generate *symmetric M-matrices* is already shown to be efficient. We discuss below some other types of problems to which AMG can successfully be applied.

A. WHY ALGEBRAIC MULTIGRID?

AMG, like multigrid, uses the same multi-scale hierarchy to solve the problem $Au = f$. At each level during the solution process, the system is "solved" and the error corrected until the process terminates at the lowest level (when no more levels are required) and then the system is solved by using one of a host of direct methods such as Gaussian elimination. Thus far, there seems to be little difference, but shortly, we will see that there is.

It is a solution method that is ideal for solving more general large, sparse algebraic equations because, unlike MG, it is not dependent on particular domains or operators. Although AMG can solve standard elliptic problems on uniform rectangular grids, for such problems, it is a less efficient solver than highly specialized geometric multigrid solvers. However, once the setup phase (discussed in the next section) is completed, AMG is quite competitive. But when geometric grids become impossible to design and MG cannot be tailored to fit the problem, this is when AMG should be used. It has been shown in various papers such as [Ref. 13, 14, 15] to have favorable results for these more complicated domains.

Even though originally designed to solve large, sparse matrix equations that involved M-matrices, application of AMG to other types of not-so-sparse matrices has increased in recent times. This is primarily due to current problems of interest which are on unstructured grids and are extremely large, usually on the order of millions of equations and unknowns. In fact, in some particular cases, algebraic multigrid has been proven to be the fastest method available. One problem, for example, is the transistor placement problem in integrated circuit design. This particular problem concentrates on layout optimization and solves a system that determines the optimal location for hundreds of thousands of cells, which consist of hundreds of transistors, on the chip surface. Regler and Rüde show that AMG is a competitive alternative to current methods in layout optimization.

B. APPLICATIONS OF AMG

Algebraic multigrid methods are useful when it comes to solving certain systems that either contain no geometric structure or that produce irregular grids tailored to complex domains or operators. When the application of standard multigrid methods proves to be impossible or entails a high degree of difficulty on certain problems, algebraic multigrid may be a better choice. These types of problems are described in more detail in [Ref. 14] but here, they are summarized. In general, AMG should be considered over geometric multigrid

(1) when the problem involves complex domains so that *any* discretization we choose is still too fine to serve as the coarsest grid. In this case, the work required in geometric multigrid to determine the interpolation schemes would be prohibitive. AMG is clearly the choice here since the coarsening process is automatic. In fact, for this case, AMG performs quite favorably in terms of operation count and CPU time [Ref. 15].

(2) when uniform coarsening is not allowed at all on the finest grid. This happens, for example, when a finite element discretization is applied using irregular triangulations. Since AMG requires no geometric structure (i.e., no uniform grids), it is ideal.

(3) when the interpolation operator is chosen so that it becomes very difficult to find a relaxation process that complements the operator in smoothing the error enough to allow a decent coarse grid correction. Sometimes, it is impossible to determine a relaxation process at all. AMG is not affected here since interpolation is defined by the entries in the matrix, and the weights are chosen tot reflect the relative strength of each connection.

(4) when a problem is entirely discrete, especially if the problem contains no geometric structure at all. Since AMG does not depend on an underlying continuous problem, it is clearly the method of choice. This applies to problems where we are only given the system matrix A and the right-hand side vector f.

C. THE STEPS OF AMG

Algebraic multigrid methods are designed with regular (geometric) multigrid methods in mind, in that they use the principles of multigrid but do not require nor rely on the geometry of the particular problem being solved. Instead, they explicitly use information from the matrix of the system, i.e., the coefficients of the unknowns. Application of AMG to the problem $Au = f$ requires a two-part process. There is an initial step which defines the intergrid transfer and coarse-grid operators and, in addition, chooses the coarse grid itself. This first step of the process is called the *setup phase* and most of the work involved in writing the algorithm takes place in this portion of AMG. "The generality of AMG must be paid for by a setup phase that may take 80% or more of the overall time [Ref. 15]". The second part of the AMG process is called the *solution phase.* In this part, regular multigrid cycling is performed on the components of the matrix until a predetermined tolerance is met.

1. The Setup Phase

As defined in [Ref. 14], a brief outline of the algorithm used in the setup phase on the problem $Au = f$ is provided below:

1. Set $m = 1$.
2. Choose the coarse grid Ω^{m+1} and define I^m_{m+1}, the interpolation operator.
3. Set $I^{m+1}_m = (I^m_{m+1})^T$ and $A^{m+1} = I^{m+1}_m A^m I^m_{m+1}$.
4. When Ω^{m+1} is small enough, set $q = m + 1$ and stop. Otherwise, let $m = m + 1$ and return to step 2.

Here, we define q to be the number that represents the coarsest grid. This number is used in the *theoretical* bound on the asymptotic (V-cycle) convergence factor, ρ. In practice though, ρ is q-independent and by increasing the accuracy of interpolation for smooth errors, we find that AMG is quite an efficient method. Therefore, the large investment in writing the setup phase may save time (and cost) in the setup work for later problems that use the same, or nearly the same, matrix. This effectively reduces the time required for the solution of follow-on problems.

2. The Solution Phase

The solution phase is part of the algorithm in which regular multigrid cycling takes place. It involves a relaxation process to remove the oscillatory modes of the error and an intergrid transfer operators to move the problem to different levels. This phase of AMG, when the operators and relaxation are chosen properly, is a very efficient solver for the problem on the finest grid [Ref. 14].

D. DEFINING THE "GRID"

One of the main ideas of algebraic multigrid is to split the set of variables on a particular level into two *disjoint* subsets. The first set contains coarse level variables (C-variables) and the second is the complementary set of fine level variables (F-variables). The fact that we require the sets to be disjoint will be important as we will see shortly. If we now define h and H to be any two consecutive levels with h as the fine one then we can set $C^h = \{j \mid j \in C\}$ and $F^h = \{j \mid j \in F\}$. Furthermore, let $i \in \Omega^h = C^h \cup F^h$ be defined as a point in a fictitious plane. In this sense, i is nothing more than a reference to the variable u_i^h. Now, for example, $A^h u^h = b^h$ can be interpreted as a grid equation on a fictitious grid Ω^h. Furthermore, we define $\Omega^h = F^h$ and $\Omega^H = C^H$. The detail in defining the grid can be quite cumbersome and it is not our intention here to reproduce what can be found in [Ref. 14].

E. CONNECTIONS AND CONVERGENCE

Next we introduce a new term called *connections.* Connections refer to the relationship between grid points in the sense of a directed graph which is associated with any matrix. On any level h, a point $i \in \Omega^h$ is defined to be (directly) connected to a point $j \in \Omega^h$ if $a_{ij} \neq 0$. Furthermore, we define the (direct) *neighborhood* of a point $i \in \Omega^h$ by

$$N_i^h = \{j \in \Omega^h \mid j \neq i,\ a_{ij}^h \neq 0\}. \tag{III.2}$$

Let us now concern ourselves with interpolation along direct connections. In this case, we consider only the interpolation points C_i with $C_i \subseteq N_i \cap C$ and the corresponding weights

$$w_{ik} = \eta_i \left|a_{ik}^h\right|, \quad (i \in F,\ k \in C_i) \tag{III.3}$$

with

$$0 \leq \eta_i \leq 1 / \sum_{l \in C_i} \left|a_{il}^h\right|. \tag{III.4}$$

In order to maintain two-level convergence, it is sufficient to require for every $i \in F,\ k \in C_i$

$$0 \leq a_{ii}^h w_{ik} \leq \beta \left|a_{ik}^h\right|, \quad 0 \leq a_{ii}^h (1 - s_i) \leq \beta \sum_j a_{ij}^h \tag{III.5}$$

where $s_i \leq 1$, which is ensured by (III.4). From (III.5), we can easily derive more practical conditions to effectively develop an automatic coarsening algorithm that has β as an input parameter. The algorithm will also choose interpolation with weights (III.3) satisfying (III.5).

In order to understand the effect of β in (III.5), we state the following result presented in [Ref. 14].

> **Theorem:** Let $\beta \geq 1$ be fixed. Assume for any symmetric, weakly diagonally dominant M-matrix A^h the C-points are picked such that, for each $i \in F$, there is a non-empty set $C_i \subseteq N_i \cap C$ with
>
> $$\beta \sum_{j \notin C_i} a_{ij}^h \geq a_{ii}^h \tag{III.6}$$
>
> and define the interpolation weights (III.3) by $\eta_i = 1 / \sum_{j \notin C_i} a_{ij}^h$. Then two-level convergence is satisfied.

For a given β, condition (III.6) is satisfied by many choices of the sets C and C_i; but regardless of how we chose those sets, we always want them as small as possible. This condition requires that we make each F-point i *strongly connected* to its interpolation points. In other words, we actually want each $|a_{ij}|$ $(j \in C_i)$ to be comparable in size to the largest of the $|a_{ik}|$ $(k \in N_i)$ or equivalently,

$$|a_{ij}| \approx \gamma \max_{k \in N_i} |a_{ik}| \quad \text{for } j \in C_i \text{ and } 0 \leq \gamma \leq 1.$$

The assumption in (III.6) gets weaker as β gets larger; but, for large β, (III.6) allows for rapid coarsening. The price of this though is slower two-level convergence speed. Although, when $\beta = 1$, we achieve the best convergence but the expense of the method becomes extreme. Ruge and Stüben explain that the best compromise is when $\beta = 2$. This is when about 50% of the total strength of connections of every F-point will be represented on the coarser level.

IV. CONSTRUCTING THE INTERPOLATION OPERATOR

The choosing of an interpolation operator is "dictated by the desire to achieve good theoretical estimates of the convergence of the iterations as well as by bounds on the computational complexity of the algorithm [Ref. 11]." It is of the essence in defining the interpolation operator that the range of interpolation approximates those errors not effectively reduced by relaxation. To make it successful then, AMG must be constructed in such a way to ensure that this automatically happens.

A. ALGEBRAIC SMOOTHNESS

For the problem $Au = f$, let us define a smoothing process

$$u^m_{\text{new}} = G^m u^m_{\text{old}} + I^m - G^m (A^m)^{-1} f^m$$

where G is a (linear) smoothing operator such that $G^m : \mathcal{R}^{n_m} \rightarrow \mathcal{R}^{n_m}$. In AMG, an error e is said to be smooth if $\|Ge\|_1 \approx \|e\|_1$. In other words, the error is smooth when it is slow to converge with respect to the smoothing operator. If we now assume that e is a smooth error then the residual $r = Ae$ is small compared to e. This occurs in most of the common relaxation processes. Knowing this, we expect $|r_i| \ll a_{ii} |e_i|$ for all i and so for (algebraically) smooth error we have $Ae \approx 0$. A good approximation for each e_i can then be obtained from

$$Ae = a_{ii} e_i + \sum_{j \in N_i} a_{ij} e_j \approx 0$$

or

$$e_i = \frac{-1}{a_{ii}} \sum_{j \in N_i} a_{ij} e_j \qquad \text{(IV.1)}$$

where the neighborhood N_i of point i is defined in (III.2).

With this fact, it is easy to construct an interpolation operator which guarantees that the smooth error lies in the range of interpolation. One standard operator might be

$$\left(I_H^h e^H\right)_i = \begin{cases} e_i^H & \text{for } i \in C = \Omega^H \\ \sum\limits_{k \in C^h} w_{ik} e_k^H & \text{for } i \in F = \Omega^h \end{cases}$$

with $w_{ik} = \delta_{ik}$ if $i \in C^h$. Recall that

$$\delta_{ik} = \begin{cases} 0 & \text{for } i \neq k \\ 1 & \text{for } i = k. \end{cases}$$

However, this basic "model" interpolation operator isn't very effective since it is not usually "local" enough. Here we define "local" to be $w_{ik} = 0$ for all $i \in F^h$ unless $k \in C_i^h$, where $C_i^h \subset C^h$. For reasons of efficiency, we require that C_i^h be some small set of interpolation points. When working with symmetric M-matrices, [Ref. 14] shows for the case when $\sum_{j \neq i} |a_{ij}| \approx a_{ii}$ that smooth error generally varies slowly in the direction of strong connections. This fact will be important later on when constructing weight definitions w_{ik} for the interpolation operator along those connections.

B. INTERPOLATION ALONG DIRECT CONNECTIONS

Let us consider only those interpolation points C_i such that $C_i \subseteq N_i \cap C$. The interpolation weights then have the form $w_{ik} = \eta_i \left|a_{ik}^h\right|$ for $i \in F$ and $k \in C_i$ where $0 \leq \eta_i \leq \left(\sum_{l \in C_i} \left|a_{il}^h\right|\right)^{-1}$. We require this definition to ensure that $w_{ik} \leq 1$. Let us take a look at the case when the matrix A is strictly an M-matrix (with no assumption on symmetry), i.e., $a_{ii} > 0$ and $a_{ij} \leq 0$ for $i \neq j$. If we set $C_i = N_i$, then (IV.1) gives

$$e_i = \sum_{j \in N_i} \frac{|a_{ij}|}{a_{ii}} e_j. \qquad \text{(IV.2)}$$

Now we choose $w_{ik} = \left|a_{ik}^h\right| / a_{ii}$. However, this would imply that $C = \Omega$ but we want C_i to be small; so generally, we desire the set $D_i \equiv N_i - C_i \neq \emptyset$. What we want to do is to "distribute" the non-interpolatory connections a_{ij} $(j \in D_i)$ to the interpolatory point a_{ii}.

If $|a_{ij}|$ is small, then adding a_{ij}, for $j \in D_i$, to the diagonal element a_{ii} should not hurt accuracy very much since the contribution from a_{ij} is negligible. If, however, $|a_{ij}|$ is large, then we know this replacement is reasonable since smooth error generally varies slowly in the direction of strong connections. To determine the error in this case, we begin with (IV.1).

$$\begin{aligned}
a_{ii}e_i &= -\sum_{j\in N_i} a_{ij}e_j \\
&= -\sum_{j\in C_i} a_{ij}e_j - \underbrace{\sum_{j\in D_i} a_{ij}e_j}_{\text{set } e_j\approx e_i} \\
&\approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{j\in D_i} a_{ij}e_i
\end{aligned}$$

then

$$\left(a_{ii} + \sum_{j\in D_i} a_{ij}\right) e_i \approx -\sum_{j\in C_i} a_{ij}e_j$$

or

$$\begin{aligned}
e_i &= \frac{-\sum_{j\in C_i} a_{ij}e_j}{a_{ii} + \sum_{j\in D_i} a_{ij}} \\
&= \frac{\sum_{j\in C_i} |a_{ij}|\, e_j}{a_{ii} + \sum_{j\in D_i} a_{ij}}. \qquad \text{(IV.3)}
\end{aligned}$$

Since A is assumed to be an M-matrix, the denominator in (IV.3) is positive. We now have a good representation of the error.

C. THE ITERATIVE WEIGHT DEFINITION

Let us now consider the entries in an arbitrary matrix A. Figure 12 shows a graphical relationship between various points in the matrix. We will use this illustration in our derivation of the weights. Recall our earlier assumption that $Ae \approx 0$ when e is smooth. Then, for a smooth error and for a point $i \in F$, we have

$$a_{ii}e_i = -\sum_{k\in C_i} a_{ik}e_k - \sum_{j\in F} a_{ij}e_j - \sum_{l\in N_i,\ l\in C,\ l\notin C_i} a_{il}e_l \qquad \text{(IV.4)}$$

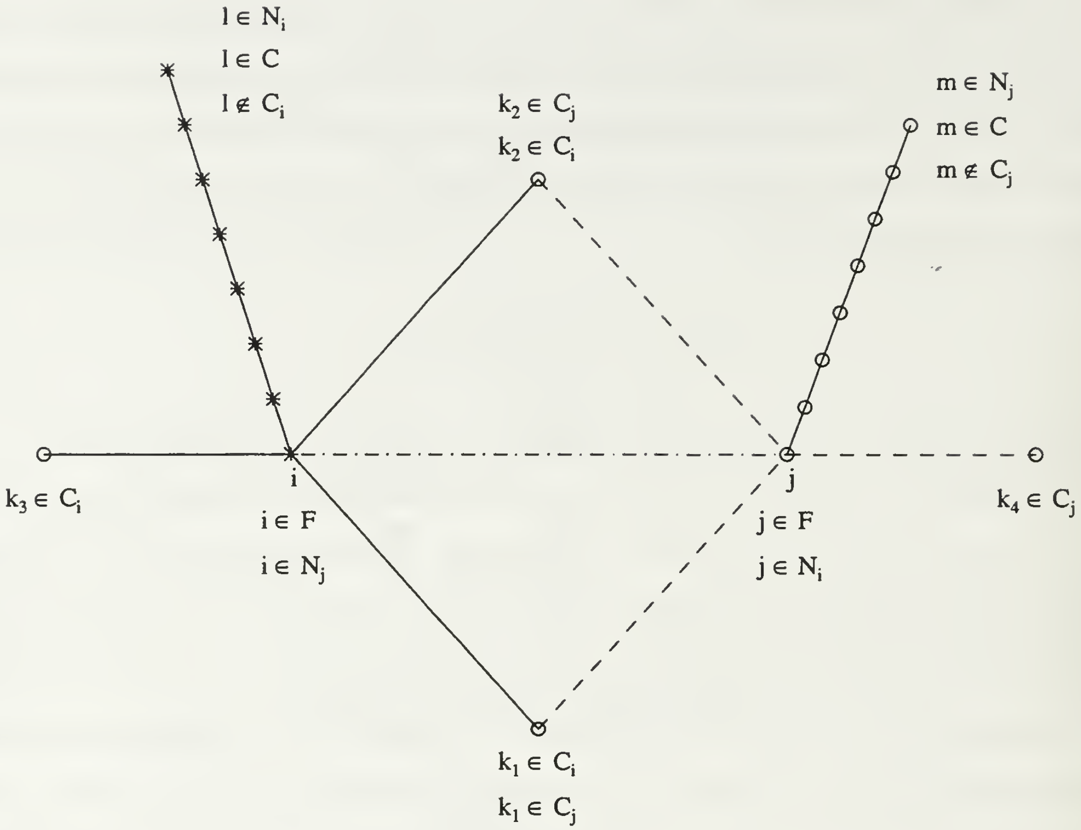

Figure 12. *Relationship of fine grid to coarse grid points in the matrix. Points i and j are the fine grid points; the others are coarse grid points, both weak and strongly connected, interpolatory and non-interpolatory.*

and similarly for a point $j \in F$, we have

$$a_{jj}e_j = -\sum_{k \in C_j} a_{jk}e_k - \sum_{i \in F} a_{ji}e_i - \sum_{m \in N_j,\ m \in C,\ m \notin C_j} a_{jm}e_m. \tag{IV.5}$$

To make the interpolation operator "small," we must distribute the non-interpolatory connections (a_{ij} for $j \notin C_i$) to the interpolatory points. In basic terms, we need to approximate e_j ($j \in F$) and e_l ($l \in N_i$, $l \in C$, $l \notin C_j$) in terms of e_k ($k \in C_i$). Likewise,we need to approximate e_i ($i \in F$) and e_m ($m \in N_j$, $m \in C$, $m \notin C_j$) in terms of e_k ($k \in C_j$).

Let us assume that the interpolation weights have been determined for the interpolatory points $k \in C_j$. That is, the interpolatory weights to define e_j for $j \in F$ are known. Then e_j could be approximated by a weighted average (for use in approximating e_i):

$$e_j \approx \frac{w_{jk_1} e_{k_1} + w_{jk_2} e_{k_2}}{w_{jk_1} + w_{jk_2}}$$

so in general

$$e_j \approx \frac{\sum\limits_{k \in C_i} w_{jk} e_k}{\sum\limits_{k \in C_i} w_{jk}}. \tag{IV.6}$$

Then for $i \in F$, we substitute e_j in (IV.6) into e_j in (IV.4). This leads to

$$a_{ii} e_i = -\sum_{k \in C_i} a_{ik} e_k - \sum_{j \in F} a_{ij} \left(\frac{\sum\limits_{k \in C_i} w_{jk} e_k}{\sum\limits_{k \in C_i} w_{jk}} \right) - \sum_{l \in N_i,\ l \in C,\ l \notin C_i} a_{il} e_l, \tag{IV.7}$$

while for $j \in F$, we replace e_i in (IV.5) by the equivalent approximation for e_i in (IV.6). This gives us

$$a_{jj} e_j = -\sum_{k \in C_j} a_{jk} e_k - \sum_{i \in F} a_{ji} \left(\frac{\sum\limits_{k \in C_j} w_{ik} e_k}{\sum\limits_{k \in C_j} w_{ik}} \right) - \sum_{m \in N_j,\ m \in C,\ m \notin C_j} a_{jm} e_m. \tag{IV.8}$$

If we look at the two previous equations closely, we see that (IV.7) is being used to define w_{ik} which are used in (IV.8) and (IV.8) is being used to define w_{jk} which are used in (IV.7). Thus, the interpolation weights at point i are dependent on the interpolation weights at point j, and vice-versa. Hence, we have an implicit system to define the interpolatory weights. This implicit system governs how the weights are computed and is the basis for the iterative weight definition (IWD) of AMG.

D. INTERPOLATION CONSTRUCTION USING THE ITERATIVE WEIGHT DEFINITION

This section presents an algorithm for coding the iterative weight definition of algebraic multigrid. We begin by starting at a fine grid point, say point i, and dividing its neighborhood, N_i, into four distinct groups. Figure 13 represents an illustration of a typical neighborhood of an arbitrary point i. It also shows the elements in each of the groups to which we will refer in this discussion of construction.

We begin the algorithm by dividing the entries of the matrix into four groups. For each point i, divide N_i into four groups:

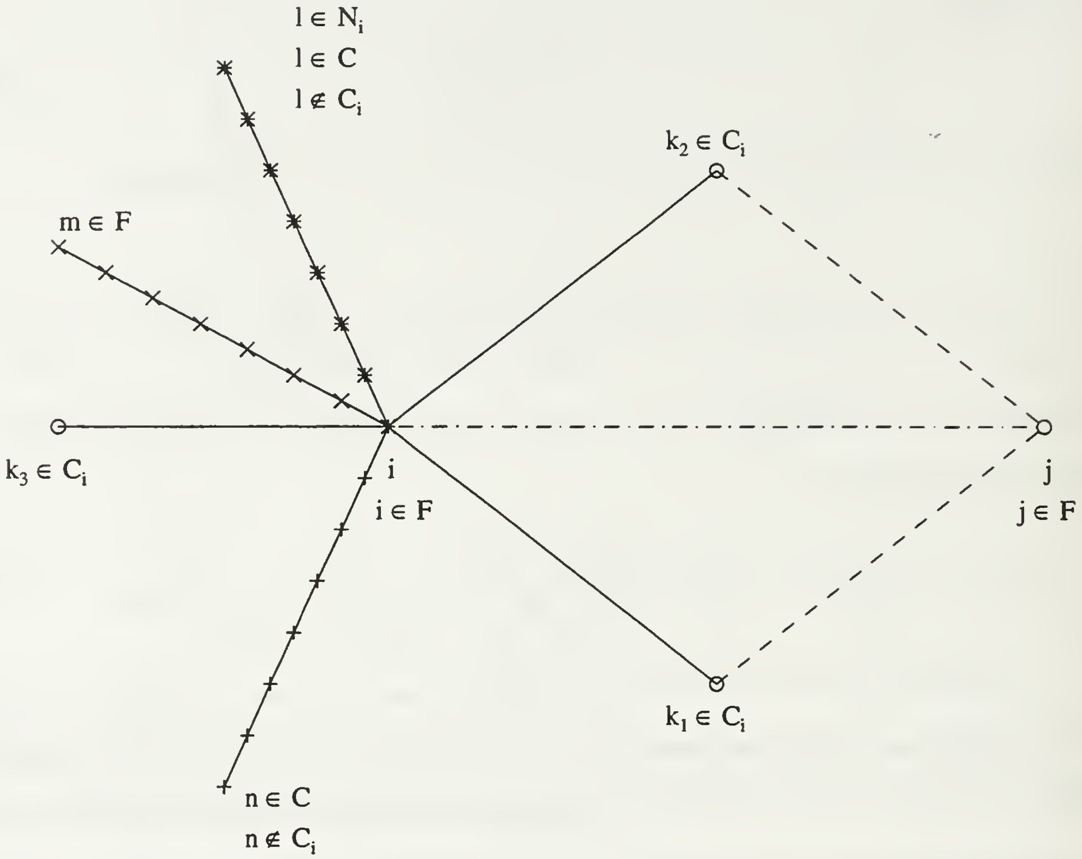

Figure 13. *Illustration of the neighborhood (N_i) of a fine grid point i and its relationship to other fine and coarse-grid points in the matrix. For each point i, N_i is divided into four distinct groups.*

Group A	Coarse-grid interpolatory points, C_i. In Figure 13, these correspond to points k_1, k_2 and k_3.
Group B	Coarse-grid and fine-grid non-interpolatory points that are weakly connected to point i. In Figure 13, these are points m and l.
Group C	Fine-grid points strongly connected to point i. In Figure 13, this is point j.
Group D	Coarse-grid points not weakly connected to point i but not in C_i. In Figure 13, this is point n.

1. Initialization

Step 1: Initialize the variables (vectors):

```
For all $i \in F$ do
    set
        $C_i = S_i \cap C$
    set
        $w_{ik} = \frac{a_{ik}}{\sum_{l \in C_i} a_{il}}$
    define
        $\Omega_i = \{ j \mid a_{ij} \text{ is small } \}$
end
```

Here, we note that S_i is the set of points strongly connected to point i and Ω_i is the set of points weakly connected to i.

2. Calculation

Step 2: Calculate the interpolatory weights, w_{ik}, by using the approximation $Ae \approx 0$:

```
For each $i \in F$ do
    $a_{ii} e_i \approx$
        $- \sum_{k \in C_i} a_{ik} e_k$                                   (IV.9)
        $- \sum_{l \in \Omega_i} a_{il} e_l$                              (IV.10)
        $- \sum_{j \in F,\ j \notin \Omega_i} a_{ij} e_j$                 (IV.11)
        $- \sum_{n \in C,\ n \notin C_i,\ n \notin \Omega_i} a_{in} e_n$  (IV.12)
    approximate the errors:
        for group B points, distribute the quantity in (IV.10) to the
        diagonal by the approximation

                $e_l \approx e_i.$        (IV.13)

        for group C points, approximate

                $e_j \approx \frac{\sum_{k\in C_i} w_{ik}e_k}{\sum_{k\in C_i} w_{ik}}$        (IV.14)

            but distribute the quantity in (IV.11) to the diagonal only
            when $w_{ik} > 0$.
        for group D points, approximate

                $e_n \approx \frac{\sum_{k\in C_i} a_{ik}e_k}{\sum_{k\in C_i} a_{ik}}$        (IV.15)

            but distribute quantity in (IV.12) to the diagonal only when
            $a_{ij}$ is of the right sign.
Either stop here, or
    For each $i \in F$ do
        set

                $C_i = \{j \mid w_{ij} > 0\}$

        goto Step 2.
    end
end
```

We give some insight to the iterative nature of this last step. The routine ends when there are no more fine-grid points or the set of interpolatory coarse-grid points, C_i, is empty. If this is not the case, we check the next fine-grid point i and refine the set C_i to include only those weights that are of the correct (positive) sign. If all the weights are now of the correct sign, then we end, if not we refine C_i again and repeat the loop until we meet an end of loop criteria.

E. INTERPRETATION OF THE ALGORITHM

In this section, we interpret the code of Section D. But before we begin, we make a few simplifications. Define the sets

$$\begin{aligned} A &= \{k \mid k \in C_i\} \\ B &= \{l \mid l \in \Omega_i\} \\ C &= \{j \mid j \in F, j \notin \Omega_i\} \\ D &= \{n \mid n \in C, n \notin C_i, n \notin \Omega_i\}. \end{aligned}$$

Now, the equation involving (IV.9)-(IV.12) becomes,

$$a_{ii}e_i = -\left(\sum_A a_{ik}e_k + \sum_B a_{il}e_l + \sum_C a_{ij}e_j + \sum_D a_{in}e_n\right). \tag{IV.16}$$

In the code, the sum over the set B becomes after substitution of (IV.13) into (IV.10)

$$\sum_B a_{il}e_i, \tag{IV.17}$$

the sum over the set C becomes after substitution of (IV.14) into (IV.11) assuming $w_{ik} > 0$

$$\sum_C a_{ij}\left(\frac{\sum_A w_{ik}e_k}{\sum_{k\in C_i} w_{ik}}\right), \tag{IV.18}$$

and the sum over the set D becomes after substitution of (IV.15) into (IV.12) assuming a_{ij} is of the right sign

$$\sum_D a_{in}\left(\frac{\sum_A a_{ik}e_k}{\sum_{k\in C_i} a_{ik}}\right). \tag{IV.19}$$

After interchanging sums in (IV.18), we get

$$\sum_A \left(\frac{\sum_C a_{ij}}{\sum_{k\in C_i} w_{ik}}\right) w_{ik}e_k \tag{IV.20}$$

and after switching the sums in (IV.19), we get

$$\sum_A \left(\frac{\sum_D a_{in}}{\sum_{k \in C_i} a_{ik}} \right) a_{ik} e_k. \tag{IV.21}$$

With (IV.17), (IV.20) and (IV.21), (IV.16) becomes

$$a_{ii} e_i + \sum_B a_{il} e_i \approx - \left[\sum_A a_{ik} e_k + \sum_A \left(\frac{\sum_C a_{ij}}{\sum_{k \in C_i} w_{ik}} \right) w_{ik} e_k + \sum_A \left(\frac{\sum_D a_{in}}{\sum_{k \in C_i} a_{ik}} \right) a_{ik} e_k \right],$$

or equivalently

$$\left(a_{ii} + \sum_B a_{il} \right) e_i \approx - \sum_A \left[a_{ik} + \left(\frac{\sum_C a_{ij}}{\sum_{k \in C_i} w_{ik}} \right) w_{ik} + \left(\frac{\sum_D a_{in}}{\sum_{k \in C_i} a_{ik}} \right) a_{ik} \right] e_k. \tag{IV.22}$$

We now rewrite (IV.22) as

$$e_i \approx \sum_A \left\{ \frac{- \left[a_{ik} + \left(\frac{\sum_C a_{ij}}{\sum_{k \in C_i} w_{ik}} \right) w_{ik} + \left(\frac{\sum_D a_{in}}{\sum_{k \in C_i} a_{ik}} \right) a_{ik} \right]}{\left(a_{ii} + \sum_B a_{il} \right)} \right\} e_k.$$

The term in the braces is w_{ik}. So our formal definition of the iterative weight definition is

$$w_{ik}^{new} = \frac{-a_{ik} - \left(\frac{\sum_C a_{ij}}{\sum_{k \in C_i} w_{ik}^{old}} \right) w_{ik}^{old} - \left(\frac{\sum_D a_{in}}{\sum_{k \in C_i} a_{ik}} \right) a_{ik}}{\left(a_{ii} + \sum_B a_{il} \right)}.$$

V. NUMERICAL RESULTS

In this chapter we present some numerical results on problems involving discretized partial differential equations as well as non-geometrically generated matrix equations using algebraic multigrid methods. As will be seen shortly, AMG methods using the iterative weight definition (IWD) may not be robust but are more efficient on certain problems than standard AMG weight definitions.

Before we begin with the results, a few definitions are in order. We define a *strong connection* as

$$|a_{ij}| = \gamma \max_{k \in N_i} |a_{ik}| \quad \text{for } j \in C_i$$

where γ is a parameter that varies: $0 \leq \gamma \leq 1$. The choice we use in our experiments to define a strong connection is $\gamma = 0.25$ which has been experimentally shown to yield good results. Most problems presented use this choice of γ; however, for some problems we test with $\gamma = 0.5$ because, in certain cases, favorable results have been found (and is another avenue of research). For those problems, it will be specifically stated what γ is. For reference purposes, (1,1) V-cycles with Gauss-Seidel relaxation and C/F-ordering of points were used in all tests. The convergence factor (ρ) was computed by

$$\rho = \left(\frac{\|R_f\|}{\|R_i\|} \right)^{\frac{1}{n}} \tag{V.1}$$

where n equals the number of cycles performed, R_f is the final residual and R_i is the initial residual. The following notation will also be used throughout this chapter:

σ^A	Operator complexity
σ^Ω	Grid complexity
$\hat{\rho}$	the asymptotic convergence factor of the 20th V-cycle

where σ^A is the ratio of the total number of nonzero entries in all the matrices to that of the number of nonzero entries in the fine-grid matrix and σ^Ω is the ratio of the total number of points on all grids to that of the number of points on the fine-grid. σ^A and σ^Ω are other measures that describe the performance of AMG. After the first few experiments we use $\hat{\rho}$ instead of ρ since the 20^{th} V-cycle is more representative of the true convergence rate.

A. ISOTROPIC PROBLEMS

We wish to establish some baseline behaviors of each method in the absence of any irregularities. We discretize the Laplace operator (∇^2) on the unit square with Dirichlet boundary conditions using several different finite difference discretizations. For each problem, we use a uniform grid with mesh size $h = 1/64$. If we let $N = 1/h$ then we create a square matrix of size $N^2 \times N^2$ (or 4096 × 4096 for our problems). The stencils we use are as follows:

$$(1)\quad \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \quad (2)\quad \frac{1}{2h^2}\begin{bmatrix} -1 & & -1 \\ & 4 & \\ -1 & & -1 \end{bmatrix},$$

$$(3)\quad \frac{1}{8h^2}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad (4)\quad \frac{1}{20h^2}\begin{bmatrix} -1 & -4 & -1 \\ -4 & 20 & -4 \\ -1 & -4 & -1 \end{bmatrix}.$$

These stencils arise naturally from the discretization of

$$\begin{aligned} -\nabla^2 u(x,y) &= f(x,y) \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{V.2}$$

To get stencil (1), for example, we replace the derivatives of

$$-u_{xx} - u_{yy} = f(x,y) \tag{V.3}$$

by second order finite differences. If we let $h_x = 1/N$ and $h_y = 1/M$, where N and M are positive integers then

$$u_{xx} \approx \frac{1}{h_x^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) \tag{V.4}$$

and

$$u_{yy} \approx \frac{1}{h_y^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right). \tag{V.5}$$

Substituting (V.4) and (V.5) back into (V.3) and with the definition $f_{i,j} = f(x_i, y_j)$, then (V.2) becomes

$$\begin{gathered} -\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} - \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_y^2} = f_{i,j} \\ u_{i,j} = u_{0,j} = u_{N,j} = u_{i,0} = u_{i,M} = 0 \end{gathered} \tag{V.6}$$

where $1 \leq i \leq N-1$ and $1 \leq j \leq M-1$.

In terms of a stencil then, we get

$$-\nabla^2 = \begin{bmatrix} & \frac{-1}{h_y^2} & \\ \frac{-1}{h_x^2} & \frac{2}{h_x^2} + \frac{2}{h_y^2} & \frac{-1}{h_x^2} \\ & \frac{-1}{h_y^2} & \end{bmatrix}. \tag{V.7}$$

If we now let $h = h_x = h_y$, then (V.7) becomes

$$-\nabla^2 = \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}$$

which is exactly what we were trying to show. The other stencils can be similarly derived.

Table I compares the results of the iterative weight definition against standard AMG weights on Laplace's equation (I.1) using the stencils above. In all tests the initial guess for $u(x, y)$ was randomly generated. On Laplace's equation using standard stencils discretized using finite differences, we note that there is little, if any, difference in ρ of the iterative method over the standard definition. As expected though, AMG produces excellent results on these model problems using either method.

Table I. *Convergence rates in solving Laplace's equation using different isotropic operators discretized using the finite difference method.*

	Standard weights			Iterative weights		
		Complexity			Complexity	
Stencil	ρ	σ^A	σ^Ω	ρ	σ^A	σ^Ω
1	0.05037	2.3714	1.7009	0.05037	2.3726	1.6990
2	0.07903	2.2475	1.6799	0.07903	2.2588	1.6797
3	0.08409	1.4162	1.3464	0.08109	1.4172	1.3440
4	0.06541	3.7120	1.9963	0.07938	3.9734	2.0522

B. THE ANISOTROPIC PROBLEM USING THE FINITE DIFFERENCE METHOD

Now that we have some baseline results of both methods in the absence of any irregularities, we wish to try a new problem,

$$\begin{aligned} -\varepsilon u_{xx} - u_{yy} &= 0 \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned} \tag{V.8}$$

where the operator, $-\varepsilon u_{xx} - u_{yy}$, is the anisotropic Laplacian. Results in this section are given for various levels of anisotropy (ε) and again we use second order finite differences to discretize the problem. This section will demonstrate the ability of AMG to tailor the coarsening algorithm to the individual problem. Other results can be found in [Ref. 9]. The domain in the following problem is again the unit square with Dirichlet boundary conditions. The problem is discretized on a uniform grid with $h = 1/64$, using the 5-point stencil

$$\frac{1}{h^2}\begin{bmatrix} & -1 & \\ -\varepsilon & 2+2\varepsilon & -\varepsilon \\ & -1 & \end{bmatrix}. \tag{V.9}$$

We will now apply both weight methods to problem (V.8) with $f(x, y) = 0$. The motivation behind this test is to have knowledge of the error. With the error known exactly, it is easier to compare the true performance of both methods. As before, the problem is discretized on a uniform grid with $h = 1/64$, using the 5-point stencil in (V.9) above. This generates a 4096×4096 matrix. The convergence rate in Table II is that of the 20th iteration ($\hat{\rho}$) vice the average defined in (V.1). In Table II, we see that the iterative weight definition produces a better convergence rate over standard AMG weight methods for all values of ε in the anisotropic problem but these improvements are utterly insignificant since they are so small. In addition, IWD does not always produce better results in terms of solver complexity or grid complexity. Clearly the standard weight definition is the better in this case since the increased cost associated with IWD makes it unfavorable when the results are practically the same.

Table II. *The anisotropic problem of the Laplacian operator discretized using the finite difference method.*

	Standard weights			Iterative weights		
		Complexity			Complexity	
ε	ρ	σ^A	σ^Ω	ρ	σ^A	σ^Ω
0.0010	0.04549	2.6319	1.9644	0.03944	2.6299	1.9609
0.0100	0.06731	2.7707	1.9590	0.06698	2.7707	1.9590
0.1000	0.05674	3.3067	1.8655	0.05672	1.3103	1.8655
0.5000	0.06349	2.4027	1.7068	0.06085	2.3272	1.6926
1.0000	0.05721	2.3714	1.7010	0.05037	2.3727	1.6990
2.0000	0.06951	2.4271	1.7129	0.05206	2.3338	1.6956
10.000	0.05576	3.5391	1.9033	0.05575	3.5691	1.9070
100.00	0.06753	2.8047	1.9658	0.06720	2.8043	1.9653
1000.0	0.04544	2.6319	1.9644	0.03946	2.6299	1.9609

Again, one can see in Table II that the differences, if any, between the methods are minimal. For these simple problems presented thus far, we see little benefit in using the iterative weight definition. In fact, if we consider the cost associated with IWD, we conclude that the method is worse since the standard weight method is a less expensive routine.

C. THE ANISOTROPIC PROBLEM USING THE FINITE ELEMENT METHOD

We present results in this section using the discretized Laplacian operator. Whereas Sections A and B involved discretizations using the finite difference method, we apply the finite element method to see if there are any improvements.

Table III provides results generated by elongating the discretized grid for Laplacian operator in the x-direction. We also compare the convergence rates of both methods as we change the definition of strong connection from $\gamma = 0.25$ to $\gamma = 0.5$. In the table, $dx = \varepsilon dy$ means that for every point in the y-direction (dy), there corresponds ε more points in the x-direction (dx). The intention of this test is to see how well the iterative method tailors to semi-coarsening to improve the convergence rate.

Table III. *The anisotropic problem with zero RHS discretized using the finite element method.*

			standard weight		iterative weight	
size ($N \times N$)		$dx = \varepsilon dy$	$\gamma = 0.25$	$\gamma = 0.5$	$\gamma = 0.25$	$\gamma = 0.5$
matrix	grid	ε	$\hat{\rho}$			
400	20	25	0.14	0.14	0.14	0.14
900	30	10	0.45	0.23	0.13	0.13
900	30	100	0.83	0.53	0.82	0.23
1225	35	50	0.25	0.14	0.15	0.14
10000	100	10	0.47	0.24	0.14	0.14
10000	100	100	0.93	0.55	0.93	0.28

From inspection of Table III, it is clear that for certain problems, the iterative weight definition is significantly better. In some cases, IWD is better by over $\frac{1}{2}$ and the extra cost associated with using this method has more than paid for itself with the large reduction in convergence rate. The natural questions then are why is it better on some and not on the others? What are the special properties of the problems in which IWD outperforms the standard weight definition? Does semi-coarsening play a role? Before we can answer these questions, we look at some other problems with domains that are much more complicated.

D. PROBLEMS WITH COMPLEX DOMAINS

In this section, we present some results for some complicated domain problems using the standard Laplacian operator

$$L = \frac{\partial}{\partial x^2} + \frac{\partial}{\partial x^2}.$$

As in Sections C, we use the finite element method to discretize the problem. Figures 14, 15 16, 17, 18 and 19 are graphic illustrations (meshes) of the problems we investigate. The size of each problem varies and is noted in the following tables. Table IV shows the results of some initial experiments on these more complicated domains. Much like the problems we have already seen, we expect that little will be gained by using the iterative weight definition since on standard Laplacian operators, regular AMG already works great.

1. The Meshes

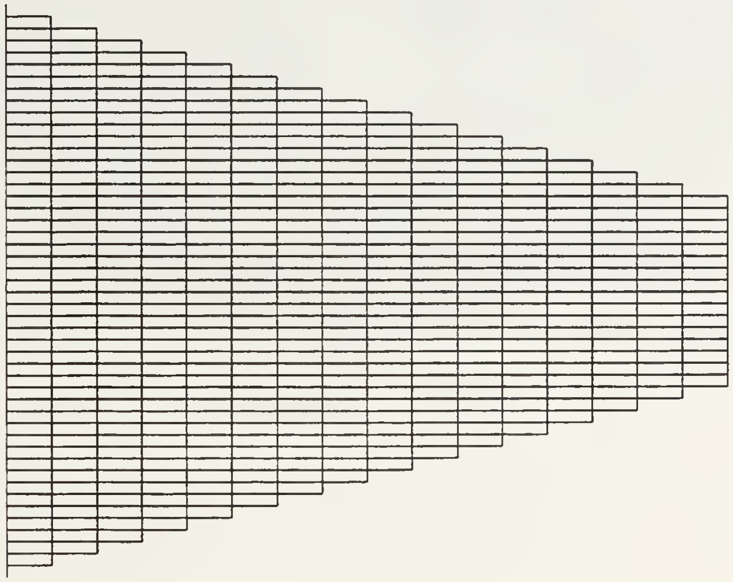

Figure 14. *Mesh 1.*

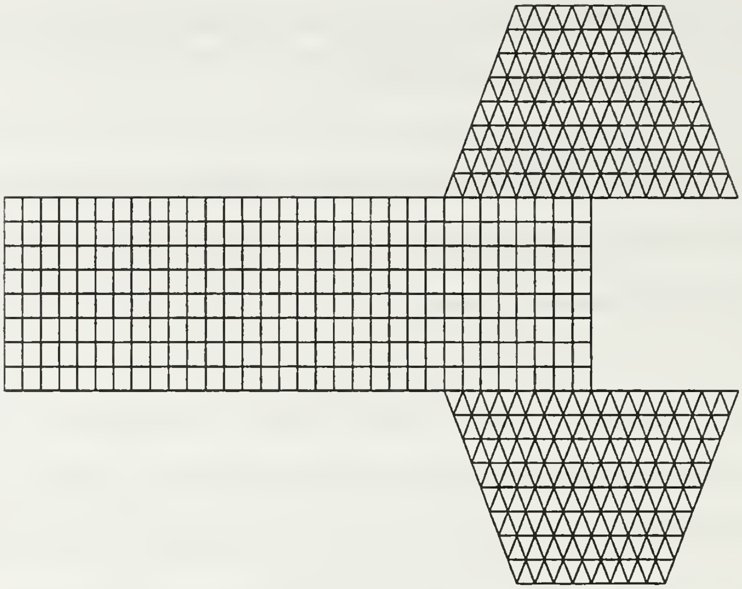

Figure 15. *Mesh 2.*

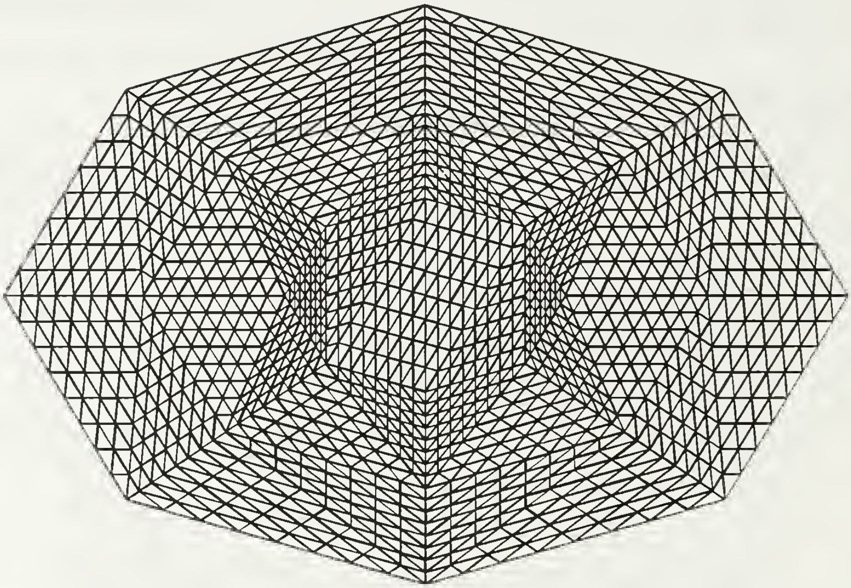

Figure 16. *Mesh 3.*

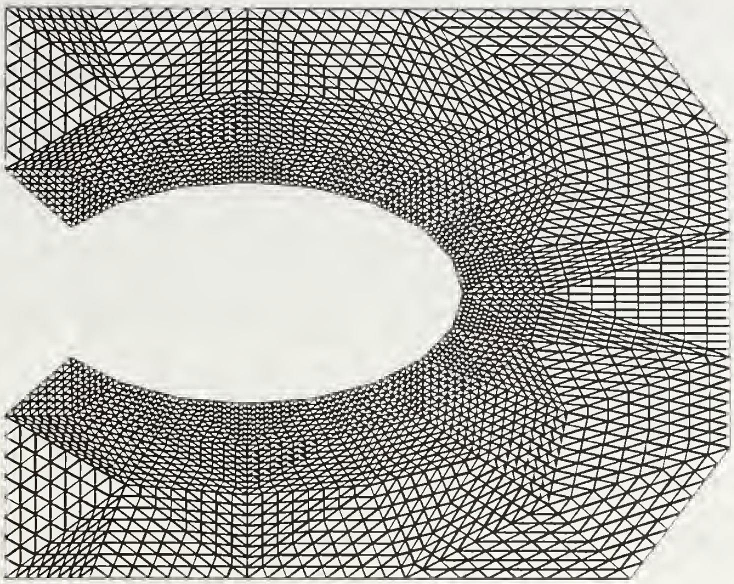

Figure 17. *Mesh 4*.

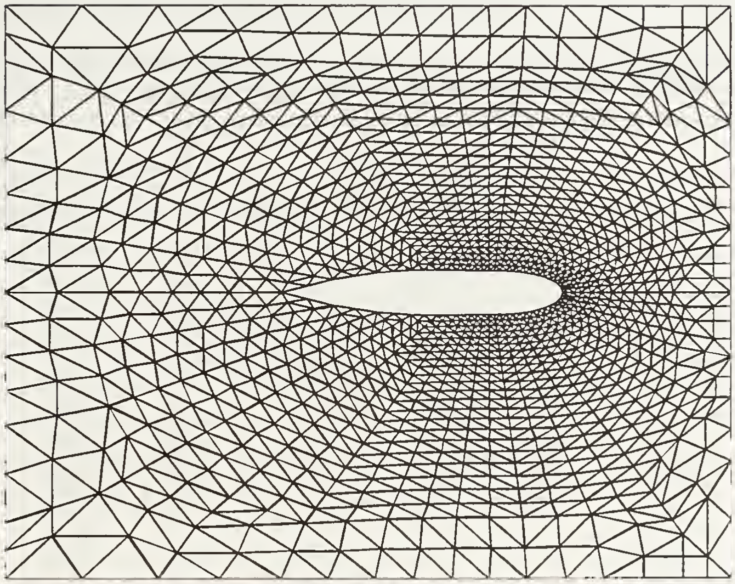

Figure 18. *Mesh 5*.

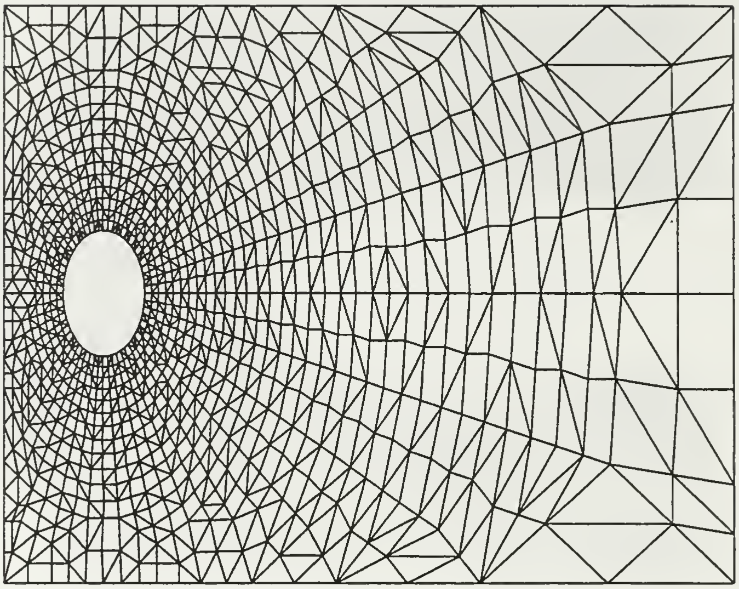

Figure 19. *Mesh 6.*

2. Results of Complex Domain Problems

Table IV. *Convergence rates of problems high in complexity using the standard Laplacian operator discretized using the finite element method.*

		Standard weights			Iterative weights		
			Complexity			Complexity	
Mesh	Size	$\hat{\rho}$	σ^A	σ^Ω	$\hat{\rho}$	σ^A	σ^Ω
1	561	0.8717	2.1990	1.7184	0.8496	2.1998	1.7184
2	513	0.0896	2.1250	1.6530	0.0902	2.1118	1.6472
3	321	0.0766	3.1268	1.9595	0.0746	3.0207	1.9315
4	757	0.1386	2.9944	1.9511	0.1779	2.9780	1.9406
5	1256	0.1460	3.3765	1.9936	0.1460	3.3000	1.9745
6	1080	0.1501	2.3064	1.7630	0.1399	2.3541	1.7769

With these results, we suppose that with simple operators such as the standard Laplacian or even the slightly skewed Laplacian the iterative weight definition will not perform better than regular AMG methods. This is largely due to the fact that conventional MG works extremely well on standard (elliptic) operators. We must test our new method using a more irregular type of operator to see if IWD will fare better than standard weight definitions.

E. COMPLEX DOMAINS AND OPERATORS

Laplacian operators have not seemed to produce results in favor of the iterative method so we hypothesize that IWD may outperform the standard weight definition method when the operator is more complex. To get a better idea of what types of operators or on what types of problems the iterative weight definition improves $\hat{\rho}$, we develop a more complex operator and reevaluate some of the same problems as in Section D.

Specifically, we experiment with those geometries as illustrated in Figures 14, 15, 17, and 16. The size of each problem here also varies and may be different from those presented in Table IV. The sizes of each problem are included in the tables. For problems in this section, we use the irregular operator

$$L = e^{x+y}\frac{\partial}{\partial x^2} + (x+y)\frac{\partial}{\partial y^2} + e^{x+y}\frac{\partial}{\partial x} + \frac{\partial}{\partial y}.$$

Table V provides results of both methods using the definition of strong connection $\gamma = 0.25$ whereas Table VI uses $\gamma = 0.5$ as the definition.

Table V. *Convergence rates of complex problems using an irregular operator discretized using the finite element method with* $\gamma = 0.25$.

		Standard weights			Iterative weights		
			Complexity			Complexity	
Mesh	Size	$\hat{\rho}$	σ^A	σ^Ω	$\hat{\rho}$	σ^A	σ^Ω
1	561	0.1367	2.3281	1.7558	0.1432	2.3891	1.7825
2	513	0.0984	2.3862	1.9415	0.1279	2.3543	1.9376
3	1249	0.5912	3.6256	2.0200	0.2877	3.6133	2.0168
4	2889	0.7949	3.4115	2.0048	0.8880	3.2519	1.9740

Tables V and VI and reveals something interesting. The iterative weight definition significantly improves the convergence rate over the standard weight definition by approximately $\frac{1}{2}$, but only in *one* case. What is special about that case? Clearly, it is not this irregular operator alone nor the specific problem but a combination of

Table VI. *Convergence rates of complex problems using an irregular operator discretized using the finite element method with* $\gamma = 0.5$.

		Standard weights			Iterative weights		
			Complexity			Complexity	
Problem	Size	$\hat{\rho}$	σ^A	σ^Ω	$\hat{\rho}$	σ^A	σ^Ω
1	561	0.2700	2.7205	1.9287	0.1194	2.7501	1.9305
2	513	0.1106	2.4271	1.9552	0.2863	2.4334	1.9591
3	1249	0.7573	3.0000	1.9976	0.4668	3.0286	2.0120
4	2889	0.9013	2.9619	2.0142	0.9400	2.9612	2.0059

both that holds the answers to our quest for classification. Moreover, in the other three problems, IWD performs worse.

Let us now investigate the case (mesh 3) where IWD works the best to see if there are any clues to its usage. Figure 20 shows not only the sparsity pattern of the matrix generated but more to the point, the locations of the entries in the matrix.

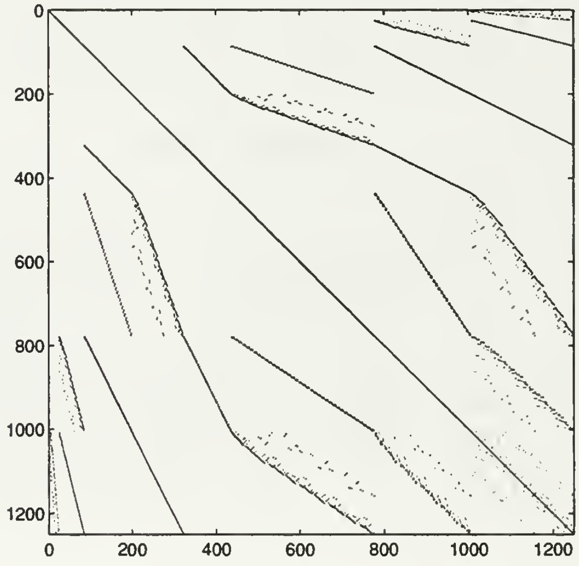

Figure 20. *The sparsity pattern of mesh 3.*

Figure 21 shows where the positive and negative entries are. Together, these figures reveal that mesh 3 is not a M-matrix and so we have a non-M-matrix case where the iterative weight definition should be used over regular AMG weight definitions.

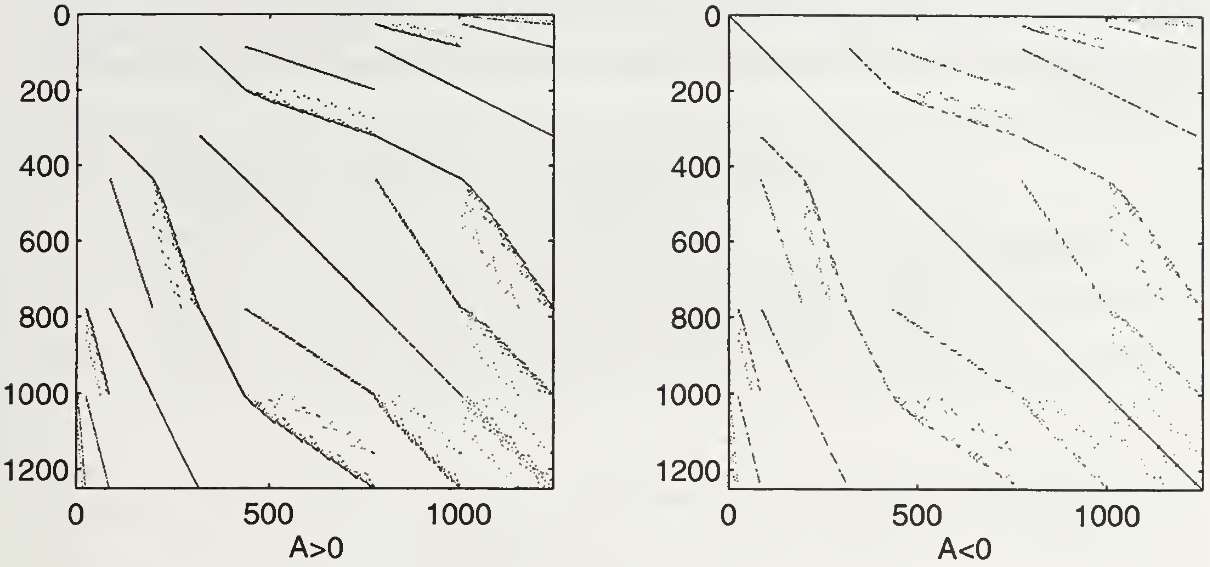

Figure 21. *Location of positive and negative entries of mesh 3.*

Table VII shows the difference between the iterative weight definition and the standard weight method over a sequence of refinements on that particular problem (mesh 3). In all cases, we see a major improvement in convergence rate.

Table VII. *Convergence rates of the problem in Figure 16 using an irregular operator discretized using the finite element method. Different refinements to the mesh are presented.*

		Standard weights			Iterative weights		
			Complexity			Complexity	
Refine	Size	$\hat{\rho}$	σ^A	σ^Ω	$\hat{\rho}$	σ^A	σ^Ω
1	85	0.3967	2.6373	1.9176	0.3411	2.7199	1.9412
2	321	0.4368	3.3284	2.0125	0.2684	3.4382	2.0467
3	1249	0.5912	3.6256	2.0200	0.2877	3.6133	2.0168
4	4929	0.7351	3.7220	2.0016	0.3582	3.6430	1.9846

Table VIII shows other experiment involving this irregular operator on the problem (mesh 5) shown in Figure 18.

Table VIII. *Convergence rates of the problem in Figure 18 using an irregular operator discretized using the finite element method. Different refinements to the mesh are presented.*

		Standard weights			Iterative weights		
			Complexity			Complexity	
Refine	Size	$\hat{\rho}$	σ^A	σ^Ω	$\hat{\rho}$	σ^A	σ^Ω
0	330	1.0000	2.9523	1.9940	0.9906	2.9927	2.0121
1	1256	0.9888	3.3843	2.0677	0.9881	3.2193	2.0239
2	4896	0.8459	3.6010	2.0666	0.8418	3.4574	2.0345

In this case, both methods have seemed to fail all together. We conclude here that the operator alone isn't the cause for the improvements seen in Table VII in using IWD. It must be a combination of both the geometry and the operator.

F. CONCLUDING REMARKS

The accuracy and efficiency of AMG in solving systems involving symmetric M-matrices has been shown in many papers. The use of AMG for solving other types of matrix equations is a topic of extensive research and variations in computational methodology during the interpolation phase have resulted in differing convergence rates. We have found that regular AMG interpolation weight definitions are inadequate for solving certain discretized systems that do not lead to M-matrices. For these types of problems, an iterative approach to determining the interpolatory weights was useful.

In applying the iterative interpolatory weight definition of AMG, we have carefully balanced the requirement of keeping the interpolatory set of points "small" in order to reduce solver complexity while at the same time, maintaining accurate interpolation to correctly represent the coarse-grid function on the fine grid. In addition, the extra work involved in using IWD does not significantly add to setup phase costs.

Experimental results have shown that the iterative weight definition does not significantly improve the convergence rate over standard AMG when applied to M-matrices, which we anticipated. However, the improvement becomes significant when solving certain types of complicated, non-standard algebraic equations although it is unclear at this stage of development what details are required to cause the iterative weight definition to outperform the regular weight definitions.

We have seen that IWD is not robust; however, there are specific problems on which IWD should be used. When using a standard operator such as the Laplacian, regular AMG should be used since its performance is superb and it is less expensive to use than the iterative weight definition we presented here. However, when the operator becomes irregular and the domain more complex, IWD has been shown to dramatically improve the convergence rate over current AMG weight definitions and should be considered a viable option in the future.

VI. FUTURE RESEARCH

Although we have shown that the iterative weight definition (IWD) of AMG isn't robust, we have found certain problems where it has improved the convergence rate significantly. We are currently searching for more data that will enable us to classify the types of problems on which IWD works best. There are still many avenues of research and IWD is just one such direction. Algebraic multigrid research is open to a plethora of new and interesting ideas. The need for more efficient and robust solvers is never-ending and is the driving force that brought about AMG in the first place. To discuss all the new possibilities of improved AMG would be a paper in itself; therefore, we restrict our attention to that of just a few new methods on the forefront of improved interpolation.

A. INTERPOLATION USING EIGENVECTORS

One method that can improve interpolation is based on the idea of eigenvector approximation and was originally presented in 1985 by Ruge and Stüben in [Ref. 14]. Approximation using eigenvectors is a method used to modify interpolation and the coarse grid matrices on all levels. The basis of the idea is that the eigenvectors corresponding to the smallest eigenvalues are, in general, the algebraically smoothest functions and so must be better approximated by the range of interpolation. Unfortunately, the eigenvector approach requires the computation of the smallest eigenvalues and their corresponding (approximate) eigenvectors. However, there are efficient methods for finding them and so it may not be as computationally expensive as one would think. Computational accuracy in determining them is rarely needed since linear interpolation (under the assumption that smooth functions are locally linear) "usually ensures accurate enough interpolation of the needed eigenvectors when standard interpolation does not [Ref. 14]."

To make the method efficient, we must integrate the computation of the eigenvector with the updating of the interpolation. Before the coarse-grid correction can be applied on each level, we must update the interpolation and the coarse-grid operator with the current eigenvector approximation. This must happen before the operation can continue to the next coarser grid. Ruge and Stüben note that this can be done efficiently and that several eigenvector approximations can be calculated simultaneously, if required. The concept of the eigenvector approximation is a tool used only for the improvement of interpolation, after which we use AMG in the standard way to solve the problem.

B. THE LEAST-SQUARES IDEA

Tom Manteuffel (University of Colorado at Boulder) and Van Henson (Naval Postgraduate School) recently introduced a new method that incorporates a least squares solution on the local level. Initial trials have produced quite favorable results. The idea behind this new method involves an singular value decomposition (SVD) of the matrix on a local level. Presented here is a brief outline of how it works.

1. Let i be a fine point. Define N_1 to be the set of points that includes the fine point i and those points p connected to point i. Furthermore, define the set N_2 to include the set N_1 and all the points q connected those points in N_1 and continue this process (e.g., $N_3, N_4, \ldots$) until there are no points remaining.

2. Now select from the matrix A the rows corresponding to N_1. Remove all columns that contain only zeros. This removal of zero-columns yields a $n \times m$ matrix which we call S. For example, if we start with a nine-point stencil on regular grid, then we end up with a 9×25 matrix, S.

3. Compute $S = U\Sigma V$, the singular-value decomposition of S. At least the last $m - n$ (or 16, using our nine-point stencil) columns of V are null-space vectors of S. Let X be the $m \times (m - n)$ matrix (e.g., 25×16) comprising these $m - n$ columns

of V.

4. Let T be the $m \times m$ matrix whose rows correspond to the same points as do the columns of S. In the example of the nine-point stencil, T is of size 25×25. T is the matrix associated with solving the homogeneous Dirichlet problem on N_2.

5. We now reorthogonalize the columns of X to be T-orthogonal, using a Gram-Schmidt process with respect to the T-inner product

$$u_j = x_j - \sum_{k=1}^{j-1} \langle Tx_j, w_k \rangle \, w_k \qquad \text{for } j = 1,\ 2, \ldots, m-n,$$

where

$$w_j = \frac{u_j}{\sqrt{\langle Tu_j, u_j \rangle}},$$

and define W be the $m \times (m-n)$ matrix whose columns are $w_1,\ w_2,\ w_3,\ \ldots, w_{m-n}$.

6. Select I, a set of k points to be the interpolatory points from which the value at i is to be interpolated. Normally (but not always) the points in I are chosen from among the C-points connected to i. Permute the rows of W so that the first k rows contain the values of the reorthogonalized singular vectors corresponding to the points in I. The $(k+1)^{st}$ row contains the values of the singular vectors corresponding to the point i.

7. Perform k Householder reflections from the *right* to bring the matrix (e.g., 25×16) W into the form:

$$\left[\begin{array}{cccccccc} * & 0 & & & & & & 0 \\ * & * & 0 & & & & & 0 \\ * & * & * & 0 & & & & 0 \\ * & * & * & * & * & * & \cdots & * \\ \vdots & & & & & \ddots & & \vdots \\ * & & & & \cdots & & & * \end{array}\right] \begin{array}{l} \leftarrow \text{ row for interpolation point} \\ \leftarrow \text{ row for interpolation point} \\ \leftarrow \text{ row for interpolation point} \\ \leftarrow \text{ row coressponding to point } i \\ \\ \\ \end{array}$$

This reflection does not alter the span of the columns, since the Householder reflection is a unitary operation.

8. Now determine a linear combination of the first k rows that gives the first k entries in the $(k+1)^{st}$ row. The weights of that linear combination are the desired interpolation weights.

Note that the size of the entries in the $(k+1)^{st}$ row beyond the k^{th} column give a "residual" measure of how well the process works. If all the entries are "small" then the weights do a good job approximating the singular vectors at point i. If some of them are "large" then we may need to add another interpolation point to the set *or*, the set of equations selected in step 2 should be expanded to be those in N_2 vice those in N_1.

Initial experiments with this method have produced excellent results on several standard problems. On certain problems requiring semi-coarsening (i.e., different coarse-grid spacings in different coordinate directions), this is the only method known to produce the "correct" interpolation weights. However, the method is extremely expensive, requiring small SVD calculations at every fine-grid point. Achieving the robustness of this approach at lower cost is a major focus of ongoing research.

C. THE COMPOSITE GRID FORMULATION

Steve McCormick (University of Colorado at Boulder) has presented some initial work on improving interpolation by approximating "globally smooth" components in the neighborhood of the interpolation region. His idea is to use *composite grids* to "solve" the local problem on the coarse grids. What he recommends is to have the target point i and all its neighbors on the local fine grid, twice removed neighbors on the next coarser and more global grid, and so on, then use the current AMG coarsening to solve the local problem which could in fact just be the least-squares scheme described in Section B. In the following discussion, L is the $n \times n$ matrix AMG is given to "invert" and I is the $q \times q$ identity matrix.

Let us assume that a point i is to be interpolated from q C neighbors. Define a composite grid consisting of the fine-grid in the interpolation region and progressively coarser grids as we move away from the target point i. We now use AMG on the composite grid with the current coarsening to compute an $n \times q$ matrix Q with the following property:

$$Q \text{ minimizes } \rho(Q^T L Q)$$

subject to

$$(PQ)^T PQ = I$$

where $\rho(A)$ denotes the spectral radius of A. Note that the $q \times q$ matrix PQ denotes the restriction of Q to the q interpolatory C points.

The motivation here is to find the vectors that lie in the near null space of L that are very distinct on the C points. Included in the space determined by Q (i.e., its span) is the "eigenvector" p of L belonging to the smallest eigenvalue where the local normalization, $(Pp)^T Pp = 1$, is used.

This scheme is reasonably cheap except that the usual coarse grids would introduce $O(\log n)$ complexity. McCormick notes that all of the coarse grids may not be needed for the interpolatory region and is hopeful that we'll need just a few very coarse ones. In this manner, we can get the interpolation we want in a fairly general setting.

If we simplify this local problem by interpreting the columns of Q to be vectors truly defined on the fine-grid then it should be fairly easy to implement. With only one target point, it's easy to see. Then, Q is an n vector. If we now partition Q into its components u, belonging to the C points, and v belonging to all other points, then the constraint $(PQ)^T PQ = I$ just becomes $p^T p = 1$.

If we partition L corresponding to the u-v partition, we have

$$\begin{bmatrix} \mathrm{A} & \mathrm{B} \\ \mathrm{B}^T & \mathrm{C} \end{bmatrix}$$

and it's can be seen that the local problem is just the eigenvalue problem:

$$\text{minimize } RQ(u)$$

where

$$RQ(u) = \frac{u^T D u}{u^T u}$$

and

$$D = A - BC^{-1}B^T.$$

Note that this determines u, and then v can be computed by $v = -C^{-1}B^T u$. Now, when we want more targets, we just solve for more eigenvectors of D. In fact, we want all q eigenvectors of the $q \times q$ matrix D. Once these eigenvectors are known, a least-squares procedure could be used to select interpolation weights, similar to the manner described in Section B. Preliminary experiments with the method outlined in this section have yielded some favorable indications, but there is much yet to be accomplished before the efficacy of the approach can be demonstrated.

The three avenues of research outlined here are all active areas of effort. All are focused on the principle that accurate interpolation of "local" null-space and *near* null-space vectors will ensure accurate approximation of the global equivalents. Moreover, an increase in accuracy of interpolation for smooth errors results in a more robust and efficient V-cycle convergence making algebraic multigrid a competitive alternative to current methods. In fact, in some cases, there is no better matrix solver. Even though this research is in its infancy, indications now point toward eventual success in using these new ideas to produce a robust, accurate, and efficient interpolation.

GLOSSARY

algebraic multigrid a numerical method used to solve particular algebraic (linear) systems with the "multigrid-style" cycling process

boundary conditions a condition imposed on a bounding surface (in three dimensions) or line (in two dimensions) or at a bounding point (in one dimension)

coarse-grid operator $A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$

connections a point $i \in \Omega^h$ is connected (directly) to a point $j \in \Omega^h$ if $a_{ij}^h \neq 0$

diagonal dominance (strict) $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad i = 1, \ldots, n$

diagonal dominance (weak) $|a_{ii}| \geq \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad i = 1, \ldots, n$

Dirichlet boundary conditions $u = 0$ on the boundary

injection one type of a linear restriction operator

iterative methods a.k.a relaxation

Fourier modes $v_j = \sin(jk\pi/N)$ where k is the wave number

Galerkin condition $A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$, known as the coarse-grid operator

interpolation operator $I_{m+1}^m : \mathcal{R}^{n_{m+1}} \rightarrow \mathcal{R}^{n_m}$

Laplace's equation $\nabla^2 u = 0$

M-matrix a matrix which is positive definite and whose off-diagonal components are nonpositive

Poisson's equation $\nabla^2 u = S, \quad S \neq 0$

restriction operator $I_m^{m+1} : \mathcal{R}^{n_m} \longrightarrow \mathcal{R}^{n_{m+1}}$

smoothing operator $G : \mathcal{R}^n \longrightarrow \mathcal{R}^n$, G acts on e to remove the oscillatory modes of the error

smoothing property the property of an iterative method that quickly removes the oscillatory modes but leaves the smooth (or low-frequency) modes of the error

strongly connected a point i is strongly connected to a point j if $|a_{ij}| \approx \max|a_{ik}|$ for $j \in C_i$ and $k \in N_i$

variational properties $I_m^{m+1} = (I_{m+1}^m)^T$ and $A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$

wave number the kth mode consists of $k/2$ full sine waves on the interval $(0, N)$

C-variables coarse level variables

F-variables fine level variables

LIST OF REFERENCES

[1] Achi Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary value problems. In *Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics*, volume 1, pages 82–89, 1973.

[2] Pieter Wesseling. *An Introduction to Multigrid Methods.* John Wiley and Sons, New York, New York, 1992.

[3] F. Krawczyk and T. Weiland. Use of a multigrid solver in the mafia module S3 for electro- and magnetostatic problems. *IEEE Transactions on Magnetics*, 26(2):747–750, March 1990.

[4] Van Henson, et. al. Multilevel image reconstruction with natural pixels. In *SIAM Journal on Scientific Computing*, volume 17, pages 193–216, Philadelphia, Pennsylvania, January 1996. SIAM.

[5] Tin-Su Pan and Andrew E. Yagle. Numerical study of multigrid implementations of some iterative image reconstruction algorithms. *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 3:2033–2037, November 1991.

[6] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. In N. D. Melson, T. A. Manteuffel, S. F. McCormick, and C. C. Douglas, editors, *Seventh Copper Mountain Conference on Multigrid Methods*, volume CP 3339, pages 721–735, Hampton, VA, 1996. NASA.

[7] William L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, Pennsylvania, 1987.

[8] Gene Golub and Charles Van Loan. *Matrix Computations.* The John Hopskins University Press, Baltimore, Maryland, 1989.

[9] G. Golubovici and C. Popa. Interpolation and related coarsening techniques for the algebraic multigrid method. In P. W. Hemker and P Wesseling, editors, *Multigrid Methods IV*, number 116 in International Series of Numerical Mathematics, pages 201–213, Basel, Switzerland, July 1994. ISNM, Birkhäuser Verlag.

[10] Achi Brandt. Multigrid techniques: 1984 guide. *Computational Fluid Dynamics Lecture Series*, March 1984.

[11] Petr Vanek, Jan Mandel, and Marian Brezina. Algebraic multigrid on unstructured meshes. Technical Report UCD-CCM-034, Center for Computational Mathematics, University of Colorado at Denver, December 01 1994.

[12] Achi Brandt, et. al. Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and Its Applications*, Cambridge, MA, 1984. Cambridge University Press.

[13] John. W. Ruge. Algebraic multigrid (AMG) for geodetic survey problems. In Steve McCormick, editor, *Preliminary Proc. Internat. Multigrid Conference*, Fort Collins, CO, 1983. Institute for Computational Studies at Colorado State Univ.

[14] John. W. Ruge and K. Stüben. Algebraic multigrid. In Steve McCormick, editor, *Multigrid Methods*, pages 73–130, Philadelphia, Pennsylvania, 1987. SIAM.

[15] Hans Regler and Ulrich Rüde. Layout optimization with algebraic multigrid methods (AMG). In *Proceedings of the Sixth Copper Mountain Conference on Multigrid Methods, Copper Mountain, April 4-9, 1993*, Conference Publication. NASA, 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road., Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code MA 1
   Department of Mathematics
   Naval Postgraduate School
   1411 Cunningham Road, Rm 341
   Monterey, Ca 93943-5216

4. Professor Van E. Henson, Code MA/Hv 2
   Department of Mathematics
   Naval Postgraduate School
   1411 Cunningham Road, Rm 341
   Monterey, Ca 93943-5216

5. Lieutenant Gerald N. Miranda, Jr. 2
   Naval Sumbarine Base
   Submarine Officer Advanced Course
   Code N222, SOAC 97060
   P.O. Box 700
   Groton, CT 06349-5700

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTE[illegible]

DUDLEY KNOX LIBRARY
3 2768 00339324 0